VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozhraní pro nativní XML databázi v prostředí .NET

.NET Interface for a Native XML Database

2010 Michal Laš

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava*.

V Ostravě 7. května 2010 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2010 .....

Rád by som na tomto mieste poďakoval všetkým, ktorí mi s prácou pomohli, predovšetkým doc. Ing. Michalovi Krátkému Ph.D. za odborné rady a vedenie počas vypracovávania tejto diplomovej práce.

Abstrakt

V dnešných dňoch sa čoraz častejšie stretávame s pojmami ako XML dokumenty, SOAP či webové aplikácie. Spoločným menovateľom týchto technológií je v neposlednom rade práve XML. S rozširovaním tohoto formátu a s postupným zväčšovaním objemu dokumentov v ňom vytváraných, vzniká potreba tieto dáta efektívne indexovať a dotazovať. Práve z tohto dôvodu začali postupne vznikať špecializované databázové riešenia, tkz. XML natívne databázy. Medzi slabé miesta týchto databází patria rozhrania pre prístup k dátam, ktoré obsahujú. Táto práca si kladie za úlohu popísať existujúce typy rozhraní a riešení, ktoré umožňujú manipuláciu s dátami. Pričom hlavným cieľom je vytvorenie sieťového komunikačného rozhrania pre jednu z týchto databází, ktorá je momentálne prístupná iba pre lokálne použitie.

Kľúčové slová: XML, XML nativní databáze, dotazovací jazyk, rozhranie, .NET, XMLDB

Abstract

We are hearing so much about XML documents, SOAP or web services in these days. The most common aspect of these technologies is the XML. With continuing expansion of this format and growing size of documents, we need efficient way of storing and quering for these data. This is the reason why the special database solutions were found, the XML native databases. One of the weaknesses of these databases is interface to access data, which are stored within. One of the goals of this thesis is to describe existing interface a solutions for manipulation with data and database itself. And the main goal is to create network interface for one of the existing databases, which is only accessible by local at the moment.

Keywords: XML, XML native database, query language, interface, .NET, XMLDB

Zoznam použitých skratiek a symbolov

API	–	Application Programming Interface
DB	–	Databáza
SOAP	–	Simple Object Access Protocol
SRBD	–	Systém riadenia báze dát
URI	–	Uniform Resource Identifier
URL	–	Uniform Resource Locator
WSDL	–	Web Service Definition Language
XML	–	Extensible Markup Language
AD	–	Vzťah typu predok - potomok (Ancestor - Descendant)
PC	–	Vzťah typu rodič - dieťa (Parent - Child)

Obsah

Seznam obrázků

Zoznam výpisov zdrojového kódu

1 Úvod

XML technológia [34] je slovné spojenie, ktoré je čoraz častejšie skloňované v mnohých oblastiach Informačných technológií. Medzi tie najčastejšie patria najmä webové služby a oblasti, kde sa často využívajú dáta ukladané vo forme stromových štruktúr. Jazyk XML bol vytvorený konzorciom W3C (The World Wide Web Consortium) [67] a hlavné dôvody jeho vývoja boli nasledujúce kritéria: možnosť dokumenty vytvorené týmto formátom čítať a upravovať v ľuboľnom textovom editore, bol ľahko rozšíriteľný a mal jednoduchú syntax. Tieto vlastnosti umožnili jeho neskoršie masívne rozšírenie. Azda najlepším dôvodom obľúbenosti XML je možnosť vytvárať si vlastné značky a tým pádom aj vlastné formáty od neho odvodené, pretože štandardne obsahuje nástroje pre ich spracovanie. Taktiež je na XML technológiu nabalené množstvo ďalších nástrojov umožňujúcich kontrolu dokumentov, ich parsovanie, vzájomné prevody a podobne. Z historického hľadiska XML naväzuje na staršiu technológiu pre vytváranie značkovacích jazykov - SGML (Standard generalized markup language) [61]. Tento štandard sa však neujal hlavne z dôvodu jeho zložitosti.

S rastúcim množstvom dát ukladaných vo forme XML vznikla potreba tieto dáta efektívne ukladať a neskôr aj dotazovať. Avšak tu nastal problém, pretože XML dokumenty sú charakteristické svojou stromovou štruktúrou vnútorného usporiadania a tým pádom nebolo možné, prípadne efektívne, ukladať tieto dáta do bežných databází založených na relačnom alebo objektovom modeli. Ako odpoveď na túto požiadavku začali vznikať špecializované databázové systémy, ktoré riešili tento problém. Tieto databázové systémy vznikali, buď ako nadstavby pre bežné relačné databázy, ktoré neskôr dáta rozdelili a potom ich ukládali do dynamicky upravovaných tabuliek relačného modelu, alebo vznikali takzvané natívne databázové systémy, ktoré už dáta ukládali podľa vlastného modelu.

Jedným zo slabých miest týchto systémov boli ich komunikačné rozhrania. Slabinou týchto rozhraní nebol ani tak výkon alebo spoľahlivosť, ale neexistencia štandardu pre ich vytváranie. Z toho dôvodu začalo vznikať veľmi veľa rôznych typov rozhraní. Obvykle to vyzeralo, tak že každá novovytvorená databáza mala svoj vlastný štyl komunikácie, teda sa pre komunikáciu vytvorilo vlastné API pre viaceré podporované jazyky. V mnohých prípadoch dokonca k jednému systému vzniklo viacero rôznych rozhraní s rôznym štýlom komunikácie. Ale mnohé databáze podporovali aj pomerne netradičné rozhrania, ako napríklad že fungovali priamo ako webové služby [65], teda mali nadefinovaný formát správ, ktoré dokázali spracovať a odpovedať na ne. Tento fakt samozrejme začal spôsobovať problémy najmä vývojárom informačných systémov a aplikácií pracujúch s natívnymi XML databázami. Objavili sa preto snahy o zjednotenie, resp. o vytvorenie štandardu. Najznámejším štandardom v tejto oblasti je pravdepodobne práca iniciatívy XML:DB [70], ktorá vytvorila podrobné programátorské API slúžiace ako šablona pre vytváranie rozhraní k natívnym databázovým systémom. Popri tomto pokuse o zjednotenie vznikali aj ďalšie riešenia ako napríklad XDBC [73], XML/DBC [69], atd.

Táto práca si kladie za cieľ zozbierať informácie o existujúcich rozhraniach k natívnym XML databázam, stručne ich popísať a na ich základe vytvoriť databázové rozhranie pre jednu existujúcu databázu, ktorá je momentálne dostupná iba pre lokálne použitie na počítači, kde je nainštalovaná. Toto rozhranie bude vytvorené v jazyku C#, pre použitie v prostredí .NET [44].

Prvá časť tejto práce (kapitola 2) je zameraná na technológiu XML, na jej popis, typy XML dokumentov a ich syntax (podkapitoly 2.1 a 2.2). Ďalej je v tejto časti uvedený popis rozširujúcich štruktúr (podkapitola 2.3), ktorými možno jednoznačne definovať schéma XML a podľa nej kontrolovať správnosť vytvorených dokumentov. Následujú dotazovacie jazyky (podkapitola 2.4) bežne používané pre dotazovanie údajov z XML dokumentov (prípadne ich kolekcií) .Na konci tejto sekcie je uvedené základné rozdelenie a popis XML databázových systémov (podkapitola 2.7) a popis rozhraní používaných pre komunikáciu s nimi(kapitola 3).

Druhá časť je venovaná implementácii vybraného rozhrania a popisu samotného vývoja (kapitola 4). Bude popísaný celý vývojový proces od špecifikácie(podkapitola 4.1), analýzy(4.3) cez implementáciu (4.4) a nakoniec testovanie (4.5). Na konci kapitoly bude výsledné riešenie porovnané s niektorými existujúcimi XML natívnymi databázami a ich rozhraniami (5).

2 Technológia XML

XML[1] (v preklade rozšíriteľný značkovací jazyk) je značkovací jazyk, ktorý bol vyvinutý a štandardizovaný konzorciom W3C. Umožňuje jednoduché vytváranie konkrétnych značkovacích jazykov pro rôzne účely a široké spektrum rôznych typov dat. XML je jazyk určený predovšetkým pre výmenu dat medzi aplikáciami a popis heterogénnych dát. Jazyk popisuje štruktúru dokumentu na základe vecného obsahu dokumentu, pre popis vzhľadu dokumentu sa napríklad v jazyku XHTML [33] používajú kaskádové štýly [29], ktoré sú definované v priloženom dokumente.

Jedným z hlavných dôvodov vzniku tohoto jazyka bol predovšetkým fakt, že v poslednej dobe vzniklo veľmi veľa štandardov a formátov dokumentov, ktoré ale nemusia byť vzájomne kompatibilné a obvykle vyžadovali špecializovaný software pre ich prezentáciu alebo spracovanie (napríklad doc, pdf, xsl . . .). Vznikol teda jednoduchý textový formát, ktorý nie je závislý na konkrétnej platforme, a je možné ho spracovávať pomocou ľubovoľného textového editoru.

2.1 Typy XML dokumentov

XML dokumenty sa delia na 2 základné typy:

1. *Dátovo-orientované XML dokumenty* - Tento typ XML dokumentov sa využíva hlavne na export a prenos dát z relačných databází. Sú navrhnuté predovšetkým pre strojové spracovanie. Príkladmi takýchto dokumentov sú napríklad: zoznam objednávok alebo vedecké dáta. Tento typ je charakterizovaný hlavne pravidelnou štruktúrou, minimálnym alebo žiadnym výskytom zmiešaného obsahu a v elementoch a atribútoch sa obvykle nachádzajú, čo možno najmenšie logické jednotky dát.

2. *Dokumentovo-orientované XML dokumenty*: Narozdiel od dátovo-orientovaných dokumentov je tento typ určený hlavne pre potreby človeka. Príkladom pre tento typ dokumentu sú napríklad knihy. Tento typ sa vyznačuje najmä málo pravidelnou alebo nepravidelnou štruktúrou a veľkým množstvom zmiešaného obsahu. Dokumentovo-orientované dokumenty sú obvykle priamo ručne písané ako XML alebo sú vytvárané za pomoci iného formátu a externého textového editora a neskôr prekonvertované na XML formát.

2.2 Syntax jazyka

Každý XML dokument pozostáva z množiny elementov, pričom každý element môže mať vnorenú ďalšiu štruktúru. Element je teda základným komponentom XML dokumentu, pričom sa jedná o text ohraničený 2 tagmi: `<start tag>` na začiatku a ukončený `</end tag>`. Vnútri elementu sa môže nachádzať čistý text, ďalšia množina elementov, ktoré

[1] Značná časť teoretickej časti bola prebratá z bakalárskej práce [10]

sú vnorené do daného elementu, atribúty, ktoré sa zapisujú ako `<názov elementu názov atribútu = hodnota>` alebo kombinácia týchto 3 jednotiek. Dátovým modelom XML dokumentu je vo všeobecnosti graf, pričom každý element je zobrazovaný ako uzol tohoto grafu a štruktúra grafu vyjadruje zanorenie jednotlivých elementov. V XML dokumentoch sa ďalej vyskytujú takzvané *metadata*, jedná sa o poznámky v dokumente, ktoré sa priamo nespracúvavajú.

Efektivita XML je vo veľkej miere závislá na štruktúre a obsahu. Aby bol dokument považovaný za správne štruktúrovaný (*well-formed*), musí mať minimálne nasledujúce vlastnosti:

- Prvý riadok obsahuje XML deklaráciu: napr. `<?xml version="1.0" ?>`.
- Musí mať práve jeden koreňový element (*root node*).
- Neprázdne elementy musia byť ohraničené štartovacou a ukončovacou značkou.
- Všetky hodnoty atribútov musia byť uzavreté v úvodzovkách - jednoduchých alebo dvojitých, pričom ale musí platiť, že jednoduchá je uzavretá. jednoduchou a naopak dvojitá dvojitou. Opačný pár úvodzoviek potom môže byť použitý vo vnútri hodnoty atribútu.
- Elementy môžu byť navzájom vnorené, ale nesmú sa prekrývať. To teda znamená, že každý element (okrem root - elementu) musí byť celý obsiahnutý v inom elemente.

Vo výpise 1 je zobrazená ukážka jednoduchého XML dokumentu, ktorý popisuje množinu osôb, u ktorých sú uvedené ich mená a adresy.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE human SYSTEM "people.dtd">
<!-- This is metadata -->
<people>
  <person gender="male">
    <name>Peter</name>
    <address>
      <street>Baker Street</street>
      <num>99</num>
      <town>London</town>
    </address>
  </person>
</people>
```

Výpis 1: Ukážka XML súboru

2.3 Jazyky pre popis dokumentu

Jazyk XML neobsahuje žiadne preddefinované elementy, preto je potrebné si svoje tagy preddefinovať. Tento krok je nutný vtedy, keď chceme neskôr kontrolovať správnosť

štruktúry vytváraného dokumentu. Pre zadefinovanie tagov a atribútov, ktoré je možné v dokumente využit'sa používajú jazyky pre popis štruktúry dokumentu (schémové jazyky). Medzi tieto jazyky patria napríklad: DTD (Document type definition) [31] a XML Schema [64].

Program, ktorý potom na základe definície schémy vykonáva kontroly správnosti sa nazýva *parser*. Medzi najznámejšie parsery patria Xerces [26] a Microsoft XML Parser [46]. Hlavnou výhodou použitia tohoto systému je možnost'automatického zisťovania väčšiny typov chýb pri vytváraní dokumentu.

Ďalšou vlastnosťou XML je, že v jednom súbore je možné používat'nezávisle na sebe viacero druhov značkovania pomocou tzv. *namespaces* [47] (menných priestorov). Táto vlastnost'umožňuje v dokumente kombinovat'niekoľko rôznych definícií bez toho, aby hrozil konflikt v pomenovávaní elementov.

2.3.1 DTD - Document Type Definition

DTD je jazyk pre popis štruktúry XML a SGML dokumentov. Obmedzuje množinu prípustných dokumentov, ktoré spadajú do danej triedy. Medzi jazyky definované pomocou DTD patria napríklad HTML [36] a XHTML.

Štruktúra triedy alebo typu dokumentu je v DTD popísaná pomocou popisu jednotlivých tagov a atribútov, ktoré sa môžu v dokumente vyskytovat'. Popisuje ako môžu byt'jednotlivé značky usporiadané a navzájom vnorené. Taktiež popisuje, ktoré elementy môžu mat'aké atribúty a pre jednotlivé hodnoty vymedzuje ich typ.

Tento jazyk sa stal obľúbený hlavne pre svoju jednoduchost', avšak nie je príliš vhodný presnejšiu definíciu XML súboru alebo zložitejšie aplikácie, pretože nepodporuje namespaces a dátové typy. Medzi jeho nevýhody patrí aj to, že DTD nie je XML súbor.

Na výpise 2 je uvedená ukážka DTD súboru, ktorý popisuje štruktúru dokumentu z výpisu 1.

```
<!ELEMENT people (person)+>
<!ELEMENT person (name,address)>
   <!ATTLIST person gender (male|female) #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (street,num,town)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT num (#PCDATA)>
<!ELEMENT town (#PCDATA)>
```

Výpis 2: Ukážka DTD súboru

2.3.2 XML Schema

Jazyk XML Schema bol špecifikovaný konzorciom W3C a je druhým najpoužívanejším schémovým jazykom (hneď po DTD). Narozdiel od jazyka DTD v XML Schema je možné používať namespaces a dátové typy elementov a atribútov, ktoré patria medzi základné (`String, integer, boolean` ...) alebo je možné zadefinovať si vlastné dátové typy. XML Schema je popísané v XSD súboroch, ktoré sú založené na XML.

XML Schema definuje:

- možnosti výskytov elementov a atribútov v dokumente
- samotné elementy a atribúty
- zanorenie, poradie a počet výskytov elementov
- či môže byť element prázdny alebo musí obsahovať text
- dátové typy elementov a atribútov

Na výpis 3 je uvedená ukážka XSD súboru, ktorý popisuje štruktúru dokumentu z výpisu 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"></xs:element>
      <xs:element name="address">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="street" type="xs:string"></xs:element>
            <xs:element name="num" type="xs:integer"></xs:element>
            <xs:element name="town" type="xs:string"></xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute  name="gender" use="required" type="xs:string"></xs:attribute>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Výpis 3: Ukážka XSD súboru

2.3.3 Ďalšie schémové jazyky

Okrem DTD a XML Schema existujú ešte mnohé ďalšie schémové jazyky, medzi ktoré patria napríklad RELAX NG [54], Schema for Object-Oriented XML (SOX) [56], Document Structure Description (DSD) [30], atd.

2.4 XML dotazovacie jazyky

XML dotazovacie jazyky slúžia na získavanie alebo úpravu častí XML dokumentov na základne nejakej podmienky. Narozdiel od klasických dotazovacích jazykov, ktoré sa používajú pri práci nad relačnými databázami (napríklad SQL) sú tieto jazyky navrhnuté pre prácu s XML dokumentami, ktoré bežné dotazovacie jazyky nedokážu popísať, keďže dátovým modelom XML dokumentov je strom.

Medzi najznámejšie XML dotazovacie jazyky patria napríklad: *XPath* [72], *XQuery* [74], ktoré slúžia najmä pre získavanie dat a jazyky *XUpdate* [80] a *XQuery Update Facility* [76], ktoré slúžia pre vkladanie, zmenu a mazanie údajov z databáze.

2.4.1 XPath

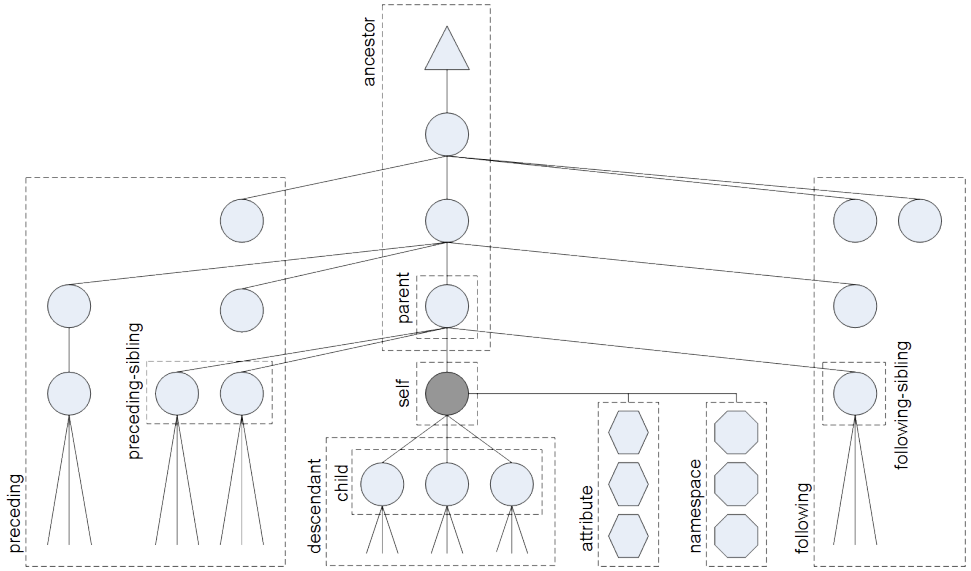
Hlavným účelom tohoto dotazovacieho jazyka je adresovanie častí XML dokumentov. Umožňuje používať základnú operácie s jednoduchými dátovými typmi ako sú: `string`, `boolean` a `čísla`. Pomocou tohoto jazyka je možné z XML dokumentu vyberať jednotlivé elementy a atribúty a získavať ich hodnoty.

Dátovým modelom XPath je strom, ktorý obsahuje 7 typov uzlov: element, atribút, root, text, metadata, menný priestor a procesná inštrukcia. Medzi jednotlivými uzlami potom môžeme prechádzať po osách. Výsledkom vyhodnotenia výrazu môže byť: množina uzlov, string, číslo alebo booleanovská hodnota.

Základnou časťou jazyka je *path expression* (výraz, ktorý popisuje cestu). Ten sa popisuje ako postupnosť prechodov medzi uzlami oddelených lomítkom. Každý prechod je potom špecifikovaný troma časťami: `os::test[predikát]` (pričom nemusia byť uvedené všetky, pokiaľ majú implicitné hodnoty):

1. *Os* (axis) - špecifikuje vzťah v strome medzi vybranými uzlami a kontextovým uzlom. Osy, ktoré sa vyskytujú v XPath: *parent*, *child*, *ancestor*, *ancestor-or-self*,*descendant*, *descendant-or-self*, *preceding*, *following*, *preceding-sibling*, *following-sibling*, *attribute*, *self* a *namespace*. Osy sú prehľadne znázornené na obrázku 1, kde síce chýbajú *ancestor-or-self* a *descendant-or-self*, ale na základe ostatných osí ich nie je zložité odvodiť.
2. *Test* (node test) - určuje typ uzlu a rozšírené meno vyberaných uzlov
3. *Predikáty* (predicates)- ktoré na základe rozhodovacích výrazov bližšie špecifikujú vybranú množinu uzlov

Vyhľadávanie potom prebieha tak, že sa postupne prechádzajú jednotlivé prechody (označované aj ako *location steps*) a z pozície kontextového uzlu sa na základe osi (smer ďalšieho prechodu) a predikátu vyberú uzly, ktoré sa v ďalšom kroku stanú kontextovými. Ako výsledok dotazu budú vrátené uzly alebo hodnoty, ktoré sa vyberú pri poslednom



Obrázek 1: XPath osy v XML dokumente

location step.

```
<suspects>
   <person gender="male">
      <name>Peter Holmes</name>
   </person>
      <person gender="female">
   <name>Alison Grey</name>
      <children>
         <person gender="male">
            <name>Thomas Gray</name>
         </person>
      </children>
   </person>
   <person gender="male">
      <name>Steve Gretten</name>
   </person>
</suspects>
```

Výpis 4: Dotazovaný dokument suspects.xml

Príklady na XPath dotazy (spúšťané na dokument z výpisu 4):

1. `/*` - získa root-element dokumentu (výsledok vo výpise 5)
2. `//person` - vyhľadá elementy s názvom *person* v celom dokumentu, bez ohľadu na zanorenie (výsledok vo výpise 6)

3. `//person[@gender="male"]/name` - vyhľadá elementy s názvom *person*, kroré majú hodnotu *male* v atribúte *gender*, následne vypíše ich meno - hodnotu elementu *name*(výsledok vo výpise 7)

```
<suspects>
   <person gender="male">
      <name>Peter Holmes</name>
   </person>
      <person gender="female">
   <name>Alison Grey</name>
      <children>
         <person gender="male">
            <name>Thomas Gray</name>
         </person>
      </children>
   </person>
   <person gender="male">
      <name>Steve Gretten</name>
   </person>
</suspects>
```

Výpis 5: Výsledok XPath dotazu 1

```
<person gender="male">
   <name>Peter Holmes</name>
</person>
<person gender="female">
   <name>Alison Grey</name>
</person>
<person gender="male">
   <name>Thomas Gray</name>
</person>
<person gender="male">
  <name>Steve Gretten</name>
</person>
```

Výpis 6: Výsledok XPath dotazu 2

```
<name>Peter Holmes</name>
<name>Thomas Gray</name>
<name>Steve Gretten</name>
```

Výpis 7: Výsledok XPath dotazu 3

2.4.2 XQuery

Tento dotazovací jazyk je rozšírením jazyka XPath. Jedná sa o komplexný dotazovací jazyk, ktorý okrem základných výrazov (obsah jazyka XPath) zahrňuje aj pokročilejšie funkcie, medzi ktoré patrí triedenie, podmienené výrazy, konštruktory atď.

Medzi základné rozšírenia dátového modelu patrí najmä možnosť práce s kolekciami dokumentov a podpora usporiadaných postupností. Ďalším rozšírením je predanie k štandardným operátorom porovnania, operátora pre uzlové porovnanie - `is`. Taktiež obsahuje množstvo vstavaných funkcií ako `min`, `max`, `exists`, `concat` ...

Veľmi podstatnou konštrukciou v jazyku XQuery sú takzvané FLWOR [74] výrazy, ktoré sú obdobou výrazov typu SELECT-FROM-WHERE z jazyka SQL. Názov tohoto typu výrazov je zložený zo skratiek názvov výrazov, z ktorých pozostáva: for, let, where, order by, return.

Na výpise 8 je uvedený jednoduchý FLWOR dotaz, ktorý vyberie z XML dokumentu z výpisu 4 všetky osoby, zoradí ich podľa mena a vráti ich ako zoznam študentov.

```
<list>
  { for $i in doc('suspects.xml')//person
    where ($i/@gender = "male")
    order by $i/name
    return
      <suspect>
        <name>{ $i/name/text() }</name>
      </suspect>
  }
</list>
```

Výpis 8: Ukážka FLWOR dotazu

Výsledok tohto dotazu je uvedený vo výpise 9.

```
<list>
  <suspect>
    <name>Peter Holmes</name>
  </suspect>
  <suspect>
    <name>Steve Gretten</name>
  </suspect>
  <suspect>
    <name>Thomas Gray</name>
  </suspect>
</list>
```

Výpis 9: Výsledok FLWOR dotazu

2.4.3 XQuery Update Facility

Rozšírením tohoto dotazovacieho jazyka je XQuery Update Facility, ktorý umožňuje modifikovať štruktúru a dáta v XML dokumentoch. XQuery Update podporuje 5 základných operácií nad XML uzlami :

- vloženie nových uzlov - `insert`.
- vymazanie jedného alebo viacerých uzlov (podľa kľúčového slova *node* alebo *nodes*) - `delete`.
- náhrada uzla sekvenciou uzlov - `replace node`.
- náhrada obsahu uzla sekvenciou uzlov alebo hodnotou - `replace value of node`.
- premenovanie uzla - `rename`.

Ukážkový dotaz XQuery Update (výpis 10) na vyššie uvedený dokument suspects.xml (výpis 4):

```
for  $gender in doc("suspects.xml")//person/@gender (: výber :)
return  (
    delete  node $gender, (: vymazanie atribútu :)
    insert  node <gender>{string($gender)}</gender> (: vloženie nového uzla :)
        as  first   into  $gender/..  ( :  ako prvého do rodičovského uzla:)
)</ list >
```

Výpis 10: Ukážka XQuery Update Facility

Výsledkom dotazu je zoznam osôb (výpis 11), kde ich pohlavie už nie je uvedené ako atribút, ale ako prvý element.

```
<suspects>
    <person>
        <gender>male</gender>
        <name>Peter Holmes</name>
    </person>
    <person>
        <gender>female</gender>
        <name>Alison Grey</name>
        <children>
            <person>
                <gender>male</gender>
                <name>Thomas Grey</name>
            </person>
        </children>
    </person>
    <person>
        <gender>male</gender>
        <name>Steve Gretten</name>
    </person>
</suspects>
```

Výpis 11: Výsledok XQuery Update dotazu

2.4.4 XUpdate

XUpdate je XML dotazovací jazyk, ktorý je vyvíjaný iniciatívou XML:DB. Tento jazyk slúži výhradne k modifikácii dát a štruktúry XML dokumentov. Operácie tohoto jazyka sú písané vo forma XML a dotazy pre výber uzlov sú riešené pomocou XPath. XUpdate funguje tým spôsobom, že na vstupe dostane XML dokument a ten je následne pretransformovaný na iný XML dokument.

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
   <xupdate:append select="/suspects">
      <xupdate:element name="person">
         <name>Mike Burton</name>
      </xupdate:element>
   </xupdate:append>
</xupdate:modifications>
```

Výpis 12: Ukážka XUpdate

Na výpise 12 je uvedený dotaz, ktorý pridáva nový záznam do XML dokumentu suspects.xml (výpis 4) pomocou príkazu *append*, atribútom *select* následne určíme miesto kam sa vloží nový záznam. Príkazom *element* potom vytvoríme nový element a ten sa vloží do dokumentu. Výsledný dokument je na výpise 13.

```
<suspects>
   <person gender="male">
      <name>Peter Holmes</name>
   </person>
   <person gender="female">
      <name>Alison Grey</name>
      <children>
         <person gender="male">
            <name>Thomas Gray</name>
         </person>
      </children>
   </person>
   <person gender="male">
      <name>Steve Gretten</name>
   </person>
   <person>
      <name>Mike Burton</name>
   </person>
</suspects>
```

Výpis 13: Ukážka XUpdate

2.5 Formátovanie a transformácie XML

Pre formátovanie a prevod XML sa využíva rodina jazykov XSL (eXtensible Stylesheet Language) [35]. Pôvodným zámerom bolo z XSL vytvoriť istú silnejšiu odrodu CSS (jazyk pre formátovanie využívaný v HTML). Postupne sa však tento zámer rozdelil do dvoch

častí : transformačná časť (XSLT) [79] a formátovacia časť (XSL-FO) [35].

XSL je tvorený troma jazykmi :

- XSLT (eXtensible Stylesheet Language Transformations) - jazyk pre prevádzanie XML
- XSL-FO (Extensible Stylesheet Language - Formatting Objects) - jazyk pre špecifikáciu vizuálneho formátovania XML
- XPath - dotazovací jazyk XML, popísaný v podkapitole 2.4.1

2.5.1 XSLT

Jazyk XSLT slúži k definícii prevodou z formátu XML do ľubovoľného iného požadovaného formátu, najčastejšie do HTML, iného XML, prípadne iných dátových štruktúr. Samotná transformácia sa potom vykonáva pomocou procesora XSTL. V dnešnej dobe existuje celá paleta softwarových procesorov, ktoré umožňujú spracovávať XML na základe definovaného XSLT. Medzi najznámejšie nástroje patria : SAXON [55], Xalan [68] a MSXML.

Pre vykonanie prevodu sú nutné 2 súbory :

- zdrojové dáta, ktoré budú transformované. Jedná sa o dáta vo formáte XML.
- vzorec pre prevod, napísaný v jazyku XSL

Pre ukážku XSLT pretransformujeme vstupný súbor z výpisu 1, ktorý je uvedený v kapitole 2.2 pomocou transformačného vzorca z výpisu 14. Uvedená transformácia zobrazí adresy všetkých osôb z Londýna v požadovanom tvare.

```
<xsl:stylesheet xmlns:xsl=' http: // www.w3.org/1999/XSL/Transform'>
  <xsl:template match="//person[address/town="London"]">
    <citizen>
      <name>
        <xsl:value−of select="name"/>
      </name>
      <address>
        <xsl:value−of select="address/street"/>
        <xsl:text> number : </xsl:text>
        <xsl:value−of select="address/num"/>
      </address>
    </citizen>
  </xsl:template>
</xsl:stylesheet>
```

Výpis 14: Ukážka XSLT transformačného vzorca

Výsledkom tohto XSLT kódu je XML súbor uvedený vo výpise 15.

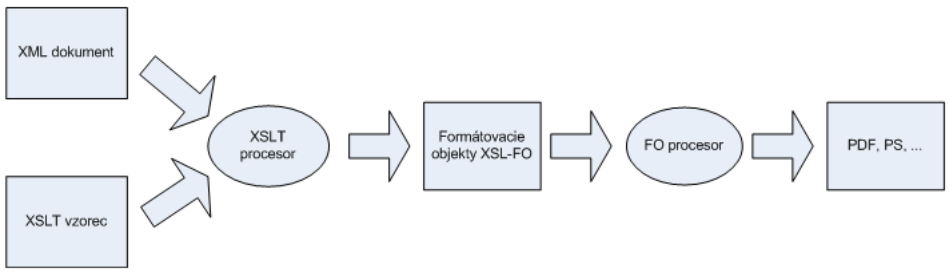
```
<citizen>
   <name>
      Peter
   </name>
   <address>
      Baker Street number : 99
   </address>
</citizen>
```

Výpis 15: Výsledok transformácie XSLT

XSLT sa v hojnej miere používa na serverovej strane, kde aplikačná logika namiesto HTML kódu generuje XML, ktoré je pred odoslaním klientovi formátované na požadovaný výstup. V prípade prehliadača je to spomínané HTML, v prípade iných typov klientov to sú rôzne iné formáty XML. Takto postavený systém je potom veľmi flexibilný z hľadiska úpravy výstupného formátu dát alebo ich ďalšieho rozširovania. Umožňuje taktiež jednoduché oddelenie aplikačnej (generuje XML) a prezentačnej vrstvy aplikácie (transformácie XSLT).

2.5.2 XSL-FO

Rozdiel oproti známej kombinácii CSS a HTML je hlavne v tom, že uživateľ nepíše priamo dokument v XSL-FO, ale v ľubovoľnom XML jazyku, napr. XHTML, DocBook. Vytvorený dokument sa prevedie pomocou XSLT do XSL-FO formátu a následne sa predá špeciálnej aplikácii (FO procesor), ktorý prevedie XSL-FO do požadovaného konečného formátu. Obvykle sa jedná o PDF, PS, prípadne iný zobrazovací formát.



Obrázek 2: Postup prekladu pomocou XSL-FO

Dokument XSL-FO nešpecifikuje presný vzhľad dokumentu, ale popisuje, aké vlastnosti má stránka mať a kam majú byť umiestnené jednotlivé jej časti. Vďaka tejto vlastnosti môžu niekedy vznikať problémy pri použití rôznych FO procesorov. Napríklad rôzne nastavené delenie slov, formát tabuliek a podobne.

XSL-FO bol navrhnutý pre stránkované média, v čom je zásadný rozdiel oproti CSS a HTML, ktoré boli navrhnuté pre súvislé média. Z tohto dôvodu sa hodí pre takzvaný *obsahom riadený dizajn*, ktorý sa využíva pre sadzbu kníh, článkov, ... Jeho protipólom je *vzhľadom riadený dizajn*, ktorý prevláda najmä v novinách (hlavný je vzhľad stránky). Pre tento typ sa FO príliš nehodí, pretože nezvláda niektoré jeho prvky.

Bez ohľadu na charakter vstupného jazyka sú schopnosti XSL-FO veľmi široké. Jeho zobrazovacie vlastností sú porovnateľné so vzhľadovými schopnosťami CSS. Umožňuje definovať tabuľky, zoznamy, plávajúce objekty, viacero stĺpcov na stránke, . . .

2.6 Indexovanie XML dokumentov

Indexovacie techniky[2] možné rozdelit'na základe typov dotazov, ktoré je možné vykonať jedným vyhľadaním v indexe na:

- Index obsahu (summary index),
- Index štrukturálneho spojenia (structural join index),
- Sekvenčne založený index (sequence-based).

2.6.1 Index obsahu

Indexy obsahu indexu atribúty a elementy na základe ciest k týmto objektom v XML dokumente, ktoré ich jednoznačne určujú. Takže jednoduchý výraz obsahujúci iba vzťahy PC, môžu byť vyhodnotené iba jedným prechodom indexom. Na druhej strane hlavne rozvetvujúce sa (*twig queries*) dotazy si vyžadujú ďalšie spracovanie, pretože často krát musia byť rozložené do niekoľkých nezávisle vykonávaných dotazov, pričom výsledky sa nakoniec zlúčia dohromady. Tento prístup nedokáže efektívne efektívne spracovávať vyhľadávacie dotazy obsahujúce vzťahy typu AD. Taktiež často zlyhávajú pri výberových podmienkach na vnútorných uzloch stromu. Vo svojej najjednoduchšej podobe, tento index priradzuje cestám v dokumente množinu elementoch, ku ktorým sa možno po tejto ceste dostať. Môže byť preto jednoducho implementovaný ako strom, v ktorom každý uzol reprezentuje názov tagu a je asociovaný so všetkými pozíciami elementov, ktoré daná cesta dosahuje od koreňa stromu po daný tag.

Indexy obsahu sa delia na 2 základné skupiny:

1. Úplné indexy obsahu,
2. Čiastočné indexy obsahu.

[2]Klasifikácia indexovacích techník bola prebratá z článku [2]

2.6.1.1 Úplné indexy obsahu

V týchto indexoch sú všetky cesty k dátam reprezentované ako strom. Tento typ indexov je vhodný pre dotazy výberu, kde sa podmienka vyskytuje na poslednom uzli, pretože v tomto prípade môžu byť dotazy vykonané pri jednom prechode indexom. Avšak tieto indexy iba zriedkavo ponúkajú pre *twig queries*, alebo dotazy s podmienkami na vnútorné uzly, a v prípade, že ponúkajú, je nutné ďalšie spracovávanie. Tieto indexy vysledujú všetky cesty začínajúce v koreni, čo ich robí vhodnými pre riešenie dotazov typu PC, avšak nie sú príliš vhodné pre dotazy so vzťahmi typu AD. Boli však navrhnuté zložitejšie dátové štruktúry, ktoré riešia niektoré tieto obmedzenia.

Hlavnou nevýhodou tohoto prístupu je jeho pamäťová zložitosť. V niektorých prípadoch rastie miesto, ktoré index zaberá exponenciálne oproti veľkosti databáze.

Príklady:

- DataGuide [5] - bol to prvý vyvinutý obsahový index. Každá cesta je asociovaná s množinou elementov, ktorými prechádza.
- Forward and Backward Index (dopredný a spätný index) [18] - je minimálny index, ktorý pokrýva všetky *twig queries*.

2.6.1.2 Čiastočné indexy obsahu

Pretože nie vždy nás zaujímajú všetky cesty databáze, je možné vytvoriť čiastočné indexy, ktoré obsahujú iba podmnožinu najbežnejších ciest, ktoré sa vyhľadávajú najčastejšie, čo umožňuje podstatne obmedziť pamäťovú záťaž. Čiastočné indexy obsahu majú obvykle rovnaké nevýhody a problémy ako úplné obsahové indexy, s ohľadom na oblasti rozvetvovania a obsahovo založených dotazov.

Príklady:

- Template Index (index šablon) [11] - podporuje reprezentáciu ciest získanú konkretizáciou nejakej špecifickej šablóny cesty. Index si ukladá triedu ciest, ktorá zodpovedá vybranej šablóne
- Adaptive Path Index for XML Data (Prispôsobivý index ciest pre XML data) [4] - tento index zvažuje najpoužívanejšie cesty a efektívne rieši dotazy pre vzťahy typu AD.

2.6.2 Index štrukturálneho spojenia

Index štrukturálneho spojenia indexuje atribúty a elementy s konkrétnymi menami alebo tie, ktoré spĺňajú dané podmienky. Obvykle sú tieto indexy vývojármi využívané pre

spracovávanie na základe spájania, pri ktorom sa najskôr vyhodnotia elementy, ktoré jednotlivým uzlom dotazu. Získané množiny sa potom spoja pomocou algoritmu *structural join* s použitím špecifickej vyhľadávacej techniky pre vylepšenie spracovania. Štrukturálne spojenie je kľúčovým problémom pre zefektívňovaní spracovávania XML dotazov a vývojári vyvinuli viacero techník, od variácií na relačné spojenie z relačných databází až po techniky redukovania výpočtu zbytočných medzivýsledkov - spoliehaním sa na použitie indexov obsahu. Vývojári môžu využiť spracovávanie na základe spájania pre riešenie oboch typov dotazov - cesty aj *twig queries*, rovnako ako aj dotazy obsahujúce typy vzťahov AD aj PC bez zmeny v spracovávaní alebo výkone.

Index štrukturálneho spojenia delíme na 3 základné skupiny:

- Jednoduché,
- Full-textové,
- Indexy založené na štruktúre.

2.6.2.1 Jednoduché

Tieto indexy vracajú množinu elementov a atribútov, ktoré splňajú určitú podmienku, ktorá môže byť na nich lokálne kontrolovaná. Táto podmienka môže zahŕňať obsah asociovaný s atribútom alebo elementom (jedná s o index založený na hodnote - *value-based index*) alebo asociovaný s ich názvom tagu (menný index - *name index*). Vývojári väčšinou využívajú jednoduché indexové techniky, využívané pri relačných databázach, ako napríklad B-stromy [28].

Príklady:

- XML Indexing and Storage System [5]

2.6.2.2 Full-Textové

Tento typ indexov vracia množinu elementov, ktoré spĺňajú určitú podmienku nad ich textovým obsahom. Full-textové indexy sa väčšinou spoliehajú na takzvané invertované indexy, pri ktorých sa každému slovu v texte priradí jeho pozícia v dokumente. Väčšinou sa tento typ indexu implementuje ako B-strom. Hľadacia technika, ktorú využívajú, umožňuje zmeniť počet dotazov na full-textové vyhľadávanie, ktoré môže aplikácia vykonať. Taktiež niektoré schémy nepodporujú tkz. hľadanie v blízkosti (vyhľadávanie skupiny slov v určitej blízkosti od seba).

2.6.2.3 Indexy založené na štruktúre

Tieto indexy vracajú elementy podľa ich štrukturálnych vzťahov (napríklad AD alebo

PC). Aplikácie môžu napríklad preskočiť predkov alebo potomkov, ktorí sa nezúčastňujú na spojení. Niektoré indexy založené na štruktúre garantujú dobrý výkon pri zmenách, aj keď spoliehajú na pozične založené techniky.

Príklady:

- XML Region Tree [6] - redukuje počet vrátených elementov, pričom záleží na požadovanom vzťahu.
- The Boxes methods [14] - využíva pozične orientovanú označovaciu schému a ad hoc dátové štruktúry pre dosiahnutie dobrého výkonu.
- Lazy Join [3] - vykonáva úpravy v dávkach a modeluje XML dokumenty ako strom XML segmentoch a potom ich zindexuje.

2.6.3 Sekvenčne založený index

V sekvenčne založených indexoch sú XML dokumenty a *twig queries* reprezentované ako sekvencia, a odpoveď na dotaz sa vykonáva pomocou kontrolovania zhody sub-sekvencií. Narozdiel od iných prístupov sekvenčne založené XML indexovanie využíva strom ako základnú dotazovaciu jednotku, a tak sa vyhýba potrebe rozoberať štruktúrovaný dotaz na niekoľko poddotazov. Podpora dotazov je takmer rovnaká ako u indexov obsahov, avšak rozdiel je v tom, že sekvenčne orientované priamo nepodporujú dotazy s výberovou podmienkou na vnútorné uzly. Problémom tohoto prístupu je generovanie falošných nálezov, to znamená, že subsekvencia nie vždy korešponduje s nálezom na podstrome medzi dotazom a dokumentami. V takých prípadoch je nutné pridať krok, ktorý odstráni tieto falošné nálezy. Vývojári už definovali aj špecifické subsekvenčné metódy aby sa vyhli tomuto problému.

Príklady:

- Virtual Suffix Tree [16] - popisuje XML dokumenty a dotazy ako sekvenciu párov, každý reprezentujúci uzol a cestu (vrátane obsahu uzla), ktorá k nemu vedie, podľa prechádzania stromom typu pre-order.
- Prüfer sequences [12] - popisuje XML dokumenty a dotazy ako sekvenciu označení korešpondujúcich k Prüferovým sekvenciám [53].
- Wang/Meng method [17] - využíva triedy sekvenčných metód

2.6.4 Ďalšie prístupy pre indexovanie XML dát

V [1] bol uvedený takzvaný *holistický prístup* pro vykonávanie XML dotazov. Na rozdiel od prístupov využívajúcich štrukturálne spojenie, holistický prístup umožňuje spracovať každý element pri vykonávaní XML dotazov najviac raz. V [9, 8] bol uvedený prístup indexujúcí všetky cesty od koreňa k listom vo viacrozmerných dátových štruktúrach [13].

Tento prístup patrí medzi tzv. *prístupy založené na cestách* spolu s prístupmi [19, 20]. Tento prístup umožňuje vykonávať dotazy obsahujúce jednu cestu s hodnotou bez nutnosti aplikovať operáciu spojenia. V článku [7] bola publikovaná optimalizácia založená na cene operácií spojenia umožňujúca efektívne vykonávanie rozvetvených dotazov(*twig queries*), tedy dotazov, ktoré obsahujú viac ciest.

2.7 XML databáze

V dnešnej dobe sa pre efektívnu prácu s XML dokumentami využíva niekoľko typov databázových systémov, medzi základné patria:

- *XML-enabled databázové systémy* - Tieto, obvykle relačné databázy, pracujú ako rozhranie pre transformáciu XML dat na ich vlastný vnútorný dátový model. V tomto type XML databází sa obvykle dokumenty najskôr rozdelia a potom sa ukladajú do dynamicky vytváraných tabuliek relačných databází. Takýmito databázami sú napríklad Oracle [51], Microsoft SQL Server [60], IBM DB2 [37].

- *Natívne XML databázové systémy* - Tieto databázové systémy ukládajú a vyberajú dokumenty na základe ich vlastného dátového modelu. Jediné rozhranie je založené na XML technológiách (ako napríklad SAX [43], DOM [63], XPath, XQuery, atd). Medzi tento typ databázových systémov patria napríklad: Berkeley XML DB [50], Xindice [25], Tamino [62], MonetDB/XQuery [45], eXist [32], Sedna XML DB [57], atd.

- Existuje ešte množstvo ďalších podporných systémov,ako napríklad: Middleware, XML Data Binding, atd. Tieto, ale obvykle tvoria iba akúsi nadstavbu iných riešení, takže sa nedajú považovať za plnohodnotné databáze.

Pretože XML-Enabled systémy sa väčšinou spoliehajú na relačné databáze, ich výkon je obvykle spoľahlivý a efektívny. Avšak pri štruktúrovanej reprezentácii, je nutné obsah dokumentu rozdeliť na niekoľko celkov podľa základného dátového model, čo môže následne znížiť celkový výkon pri dotazovaní týchto dat. Naproti tomu XML natívne databázové systému ukládajú dokumenty na základe ich vlastných techník, čo umožňuje vysokú flexibilitu a úsporu miesta. Avšak majú niekedy problémy s výkonom a efektivitou pri plnej podpore dotazovacieho jazyka XQuery, kvôli riešeniu tohoto problému býva niekedy implementovaná iba časť špecifikácie tohoto jazyka. Dnešné natívne systémy, však vo väčšine prípadov ponúkajú iba ukladanie a dotazovanie, čím majú ešte ďaleko od funkčnosti tradičných typov systémov.

Napriek tomu, že vývojári definovali v nedávnej dobe množstvo sofistikovaných indexovacích techník, nie sú tieto techniky dnešnými databázami príliš využívané. Oba typy databázových systémov sa tak spoliehajú na základné riešenia. V XML-enabled systémoch sú väčšinou jednoduché štrukturálne indexy, založené na B-stromoch. A podobne sú na tom aj XML natívne riešenia, kde sa vo väčšine prípadov indexujú iba elementy,

obsah atribútov a názvy tagov. V oboch prípadoch je ešte obvykle implementovaný full-textový index (ide o invertovaný index) pre indexovanie textového obsahu a ciest.

U XML-enabled je tento fakt daný tým, že pre vývojárov je oveľa jednoduchšie implementovať jednoduché a full-textové indexy štrukturálneho spojenia pomocou B-stromov a textových indexov, pretože tieto sú často krát podporované základnou architektúrou (obvykle relačnou).

3 Rozhrania XML natívnych databází

Komunikačné rozhrania databází predstavujú nástroj pre prístup k dátam obsahnutých v natívnych XML, respektíve databázach vo všeobecnosti. API umožňujú vytvárať aplikácie pre ich ukladanie, získavanie, modifikovanie a dotazovanie. V prípade bežných relačných databází sa rozhrania postupne štandardizovali do podoby JDBC [40], OBDC [48]. Tieto API sú podporované veľkým množstvom databázových systémov a programovacích jazykov. Keďže XML databáze sú ešte relatívne mladé oproti ich relačným súrodencom, nestačil ešte vzniknúť štandard tohto typu. V poslednej dobe sa síce objavilo niekoľko pokusov o zjednotenie, ale ešte stále v mnohých prípadoch majú databáze svoje vlastné rozhranie, ktoré je obvykle iné ako u konkurencie. Tento fakt spôsobuje nemálo problémov najmä vývojárom, ktorí sa musia týmto API neustále prispôsobovať.

V následujúcich podkapitolách budú popísané existujúce prevládajúce typy komunikačných rozhraní natívnych XML databázových systémov.

3.1 XMLDB API - XAPI

XMLDB API je vyvíjané od roku 2000 iniciatívou XML:DB.org, ktorá sa zameriava najmä na projekty spojené s manipuláciou a dotazovaním XML dát v natívnych databázach. Medzi jej ďalšie známe projekty patria: dotazovací jazyk XUpdate a jazyk SiXDML (Simple XML Data Manipulation Language) [71], ktorý slúži na definíciu a modifikáciu XML. Toto API sa postupom času stalo jedným z najpoužívanejších a je implementované hlavne v open-source projektoch, akými sú napríklad databáze : Xindice, eXist [32], MonetDB [45], Ozone [52], Sedna [57], atd. Ale v hojnom počte sa vyskytuje aj v komerčných riešeniach, napríklad v databázi Tamino.

Základné vlastnosti:

- *Driver* - Základom rozhrania je takzvaný driver, v ktorom je implementovaná prístupová logika k danej databázi. Driver je dodávaný vždy vývojármi danej databáze a implementuje rozhrania definované v XAPI. Koncept je prakticky rovnaký ako pri rozhraní JDBC.
- *Collection* (Kolekcia) - Kolekcie v XML databázach označuje kontainer, kde sa ukladajú vkladané XML dáta. Tieto konštrukcie by sa dali prirovnať k tabuľkám relačných databází. Kolekcie sú v XAPI reprezentované rozhraním *Collection*, pričom sa predpokladá, že v každej databáze existuje aspoň jedna kolekcia, kde sa dajú vkladať dáta. Z kolekcií je možné vytvárať stromovú štruktúru, to znamená, že obvykle existuje určitá koreňová kolekcia, ku ktorej je možné pripojiť potomkov.
- *Service* (Služba) - XMLDB API bolo navrhnuté s ohľadom na flexibilitu a rýchlu rozšíriteľnosť. Táto vlastnosť je dosianutá pomocou takzvaných *služieb*, ktoré tvoria

veľkú časť funkcionality rozhrania. Medzi základné služby, ktoré musia byť implementované driverom, patria : XPathQueryService, XUpdateQueryService (služby pre dotazovanie a modifikáciu dát) a CollectionManagementService (správa kolekcií dát), atd. Vďaka tejto vlastnosti je možné jednoducho doplniť podporu pre ľubovoľný ďalší dotazovací jazyk (XQuery, atd.), prípadne špeciálne služby danej databázy (napríklad TransactionService)

- *Resource Abstraction* (Abstrakcia zdrojov) - Keďže existuje viacero spôsobov ako pracovať s XML, používa XAPI abstrakciu pre prácu s obsahom uloženým v databáze. Vďaka tejto abstrakcii je možné k dokumentom, či výsledkom dotazov, pristupovať jednoducho ako k čistému XML textu, DOM objektu alebo SAX streamu.

- *API Core Levels* - XAPI je navrhnuté ako modulárne rozhranie, je teda nutné vývojárom nejakým spôsobom oznámiť, ktoré vlastnosti (services) sú implementované a ktoré nie. Preto boli definované takvané *Core levels*, u ktorých je definované, ktoré vlastnosti jednotlivé úrovne obsahujú. Napríklad *Core level 0* obsahuje iba základnu funkcionalitu (správa kolekcií, XPath, . . .), u *Core level 1* je už k základnej funkcionalite pridaná podpora transakcií a podobne. Jednotlivé úrovne implementácie sú definované na stránkach iniciatívy XML:DB [70].

3.2 XQuare Fusion API / XDBC

XQuare Fusion je komplexný nástroj (middleware) pre ukladanie heterogénnych dát (vrátane XML dokumentov a relačných databází). V rámci tohoto nástroja je definované relatívne všeobecné API pre komunikáciu s databázou (označované aj XML/DBC, prípadne XDBC). XDBC je implementované v 2 natívnych databázových riešeniach : Marklogic XML server [41] a Infonyte DB [39].

XDBC je podobne ako XAPI založené na implementácii konkrétneho drivera pre konkrétnu databázu. Rozdiel je v tom, že spojenie sa vytvára pomocou takzvaného mediatora, ktorému sa predá konfiguračný súbor s potrebnými nastaveniami (okrem iného aj s cestou k driveru) namiesto použitia triedy drivera. Syntax API (názvy tried a metód) sa do značnej miery podobá na syntax relačného JDBC, teda jeho použitie je pomerne jednoduché, pokiaľ má vývojár skúsenosti s relačnými databázami. Ako dotazovací jazyk sa používa XQuery, ktorý by mal byť implementovaný v plnej miere.

XDBC umožňuje používať mnoho funkcií, na ktoré sme zvyknutí z relačných databází, ako napríklad PreparedXMLStatement, ResultSet a podobné. Teda aj tu je vidieť inšpiráciu tvorcov v JDBC.

3.2.1 XCC

V neskorších verziách databázového systému Marklogic bol XDBC klient nahradený XCC klientom [42], pričom serverová časť XDBC a možnosť komunikovať cez XDBC klienta zostáva zachovaná. Avšak syntax tohto klienta sa od XDBC značne zmenila. Už nie je v

takej miere inšpirovaná JDBC, ale začína si raziť vlastnú cestu. Základom komunikácie sú takzvané *sessions*, kde sa vytvorí spojenie na databázu a na toto spojenie sa posielajú požiadavky.

3.3 XQJ

XQJ (XQuery API for Java) [75] je rozhranie vytvorené spoločnosťou Oracle, za účelom dotazovania XML dát pomocou jazyka XQuery 1.0. Podobne ako u predchádzajúcich API sa tvorcovia vo veľkej miere inšpirovali relačným JDBC. Princíp komunikácie je teda prakticky rovnaký ako u XAPI a XDBC. XQJ môžeme nájsť implementované napríklad v databázach : BaseX [27] a Sedna XML DB.

3.4 Rozhrania založené na SOAP správach

V mnohých natívnych XML databázach sa ako sekundárne rozhranie využíva rozhranie založené na zasielaní SOAP [59] správ (XML RPC - XML Remote Procedure Call [78]). Teda dané databáze de facto fungujú ako webové služby, kam sa pošle požiadavka v podobe SOAP správy vytvorenej v požadovanom tvare. Server ju spracuje a odpoveď odošle opäť ako SOAP správu. Definície formátu týchto správ bývajú dostupné ako WSDL súbor [66]. Toto riešenie je veľmi elegantné v tom, že podpora SOAP je dostupná v takmer každom programovacom jazyku. Vývojári teda implementujú rozhranie typu XAPI pre niektoré najpoužívanejšie jazyky a z tých ostatných je možný prístup cez XML-RPC.

Problémom je síce rôzny formát správ pre jednotlivé databáze, ale na druhú stranu sa tento problém rieši jednoduchšie ako používanie rôznych typov rozhraní. Z dôvodu rôznych formátov SOAP taktiež nemôžeme tento typ rozhraní úplne jednoznačne zaradiť medzi zjednocujúce API pre komunikáciu s natívnymi XML databázami.

3.5 Ostatné rozhrania

Ako už bolo spomenuté, databázové systémy obvykle implementujú viacero typov rozhraní, pričom mnohé z nich používajú API, ktoré nikde inde nenájdeme. Tieto ostatné rozhrania sú obvykle založené na princípe vytvorenia spojenia a zasialania požiadaviek, teda sú inšpirované existujúcimi riešeniami z oblasti relačných systémov.

Ďalším prípadom databází sú databáze, ktoré nemajú implementované žiadne sieťové rozhranie - embedded databáze. Tieto systémy teda slúžia iba ako lokálne knižnice, čo značne obmedzuje ich použitie v praxi. Príkladom takého systému je databáza Oracle Berkeley XML DB.

4 Implementácia rozhrania pre natívnu XML databázu

4.1 Špecifikácia

Účelom práce je vytvorenie programového rozhrania, ktoré by umožňovalo iným nadstavbovým aplikáciám komunikovať s natívnou XML databázou (API). Rozhranie by malo umožňovať nielen komunikáciu v rámci jedného stroja, ale aj prostredníctvom siete využívať databázu na vzdialenom serveri. API bude v prípade potreby jednoducho rozšíriteľné o ďalšie funkčnosť na oboch stranách (serverovej aj klientskej) a bude umožňovať jednoduchú modifikáciu pre ľubovolný iný natívny XML databázový systém.

Po preskúmaní dnešným možností a riešení poskytovanými inými reálnymi DB systémami, bolo pre implementáciu vybrané rozhranie iniciatívy XML:DB.org XAPI. Dôvodom je jeho značné rozšírenie, jednoduchá modifikácia a rozširovanie funkcionality. Komunikácia bude prebiehať pomocou zasialania SOAP správ, pre ktoré sa použije SOAP framework Apache Axis2/C [24]. Rozhranie bude ukážkovo implementované pre existujúcu databázu Oracle Berkeley XML DB [50], ktorá je v aktuálnej verzii dostupná iba v rámci počítača, kde je nainštalovaná. Teda jedná sa iba o akúsi knižnicu pre lokálne ukladanie, dotazovanie a spracovanie XML dokumentov, čo z nej robí ideálny príklad pre demonštrovanie funkcionality nového rozhrania. Ďalším dôvodom výberu práve tejto databáze je, že vďaka tomu, že u väčšiny ostatných riešení existuje sieťové rozhranie, nie je dostupné API pre komunikáciu lokálne, bez použitia siete. Muselo by sa teda využívať sieťové API (v mnohých prípadoch iná implementácia XAPI), čo by z vyvíjaného rozhrania urobilo iba akúsi zbytočnú medzivrstvu, ktorá by beh iba spomaľovala.

Základná implementovaná funkčnosť bude zodpovedať Core Level 0 v špecifikácii XML:DB. Bližšie informácie sú uvedené na stránkach iniciatívy XML:DB spolu s podrobným popisom rozhraní v JavaDoc formáte. V skrátenej forme sa jedná o:

- podporu práce s viacerými databázami (dokonca aj z rôznych databázových systémov).
- podpora autentifikácie uživateľov, v prípade aktuálneho napojenia na systém Oracle XML DB je síce táto vlastnosť nevyužitá (Oracle systém ju priamo nepodporuje), ale rozhranie bude túto podporu obsahovať.
- prácu s kolekciami. Rozhranie umožní vytvárať kolekcie v stromovej štruktúre, to znamená, že bude existovať jedna *koreňová kolekcia*. Pričom každá kolekcia bude môcť obsahovať neobmedzené množstvo *detských kolekcií*. Rozhranie bude umožňovať pre každú kolekciu vrátiť zoznam jej priamych *detských kolekcií*, prípadne vrátiť objekt s reprezentáciou zvolenej detskej kolekcie.
- prácu s jednotlivými kolekciami a dokumentami (rozhraním podporované budú XML a binárne dokumenty). Dokumenty sú v XAPI označované ako tkz. zdroje (Resources). Práca s dokumentami zahŕňa získanie zdroja z databáze, spracovanie, uloženie do databáze a možnosť získať zoznam uložených zdrojov v kolekcii.

- podpora oboch prístupov pre prácu s XML : SAX a DOM
- so zdrojmi bude možné pracovať samostatne alebo v množinách (ResourceSet). Množiny zdrojov budú podporovať základné operácie, ako napríklad vloženie zdroja (nie je definované, či je nutné kontrolovať, či set už daný dokument obsahuje, t.j. je možné vložiť viacero inštancií jednoho zdroja -> nejedná sa teda úplne o množinu), získanie zdroja podľa indexovej hodnoty a jeho odstránenie z množiny.
- ResourceSet bude taktiež umožňovať vrátiť takzvaný iterátor zdrojov, ktorý umožňuje jednoduché prechádzanie dokumentov obsiahnutých v sete, prípadne umožní vrátiť celú množinu ako samostatný XML dokument, kde *root element* bude pomenovaný `resource_set` a bude obsahovať elementy `resource`, kde atribút `id` bude označovať id zdroja a samotný zdroj bude obsiahnutý v tele elementu.

Naviac budú k funkčnosti úrovne 0 pridané nasledujúce vlastnosti a funkcie:

- správa kolekcií, ktorá umožní pridávať a odoberať kolekcie z databáze. Služba sa volá z *rodičovskej kolekcie* a vytvorí v rámci nej *detskú kolekciu* s daným menom. Odstránenie kolekcie bude fungovať na rovnakom princípe.
- dotazovanie dát bude v aktuálnej verzii riešené pomocou jazyka XPath. Výsledkom dotazu bude množina zdrojov, popisovaná vyššie. V neskorších verziách rozhrania sa môže objaviť aj modul pre podporu jazyka XQuery. XQuery síce momentálne nie je v XAPI podporované, ale vzhľadom k jeho možnostiam a rozšíreniu bude jeho podpora nevyhnutná.
- v špecifikácii XAPI nie je presne definované ako má fungovať dotaz XPath, ak kolekcia obsahuje podkolekcie. Spustenie dotazu bude teda platiť iba na aktuálnu kolekciu, bez prechádzania jej detských kolekcií.
- pre modifikáciu uložených dát sa bude používať jazyk XUpdate, pričom podobne ako u dotazovacieho jazyka je v budúcnosti pravdepodobná aj podpora dotazov vo formáte XQuery Update
- požiadavka pre ukladanie veľmi veľkých súborov. Táto vlastnosť umožní veľmi veľké XML dokumenty rozdeliť do menších častí a tie budú postupne odosielané na server, kde sa opätovne zložia a uložia do kolekcie. Hlavnými dôvodmi sú: obmedzená veľkosť pamäte na serveri aj klientovi a vyššia pravdepodobnosť prerušenia spojenia pri prenose veľkej SOAP správy.

Nefunkčné požiadavky:

- implementácia klientskej časti rozhrania v prostredí .NET, konkrétne v jazyku C#
- implementácie serverovej časti v jazyku C/C++
- platformová nezávislosť servera, bude ho možné zkompilovať a používať pod systémom Windows aj Linux

Do budúcnosti je plánovaná implementácia následovných vlastností a funkcií:

- šifrovanie, a to buď v rozsahu celého spojenia pomocou OpenSSL [49] alebo obsahov správ
- komprimácia dát pri komunikácii
- v dnešnej dobe nie je možná podpora transakcií, pretože vybraný framework nepodporuje *sessions*. V prípade, že bude táto podpora pridaná v niektorej neskoršej verzii alebo sa prejde na niektorý iný framework, bude pridaná aj podpora transakcií.
- spomínaná podpora jazykov XQuery a XQuery Update
- pridanie možnosti pre spustenie dotazu XPath aj na podkolekcie danej kolekcie

4.2 Použité technológie

4.2.1 Axis2/C

Axis2 je engine pre webové služby na princípe zasielania SOAP správ od spoločnosti Apache. Pre potreby diplomovej práce bola použitá verzia pre jazyk C. Engine môže byť použitý pre poskytovanie aj využívanie webových služieb. Axis2 umožňuje pracovať so štandardami *SOAP 1.1*, *SOAP 1.2* ako aj s webovými službami štýlu REST. Má taktiež vstavanú podporu pre MTOM [58], takže môže byť využívaný aj pre prenos binárných dát.

Kľúčové vlastnosti :

- podporuje jednostrannú (IN-Only) aj obojstrannú komunikáciu (IN-OUT).
- pre komunikáciu využíva protokol HTTP, čo umožňuje jeho jednoduchú integráciu do existujúcich riešení pre HTTP servery (Apache, IIS, ...). Umožňuje ale aj samostatné spustenie pomocou takzvaného *simple servera*.
- modulová architektúra.
- podpora pre WS-Addressing, MTOM/XOP, XPath (cez AXIOM model [23]), REST, WS-Policy, ...
- podpora protokolu TCP na oboch stranách (server aj klient).
- podpora pre WSDL, ktorá zároveň umožňuje generovať kostru aplikácie na základe WSDL dokumentu.

4.2.2 Oracle Berkeley XML DB [50]

Oracle Berkeley XML DB je XML natívna databáza postavená na Berkeley DB, ku ktorej pridáva XML parser, XML indexy a XQuery engine. Z Oracle Berkeley DB zdedila hlavne systém ukladania a podporu transakcií. Dokumenty v tejto databáze sú logicky organizované pomocou takzvaných kontajnerov (u ostatných XML natívnych databází sa označujú ako kolekcie).

Databázový systém[3] umožňuje ukladať dokumenty buď celé (nie je zasahované do štruktúry - *intact*) alebo rozložené na časti (*fine-grained storage*), podľa vnútorného modelu. Prvý spôsob je vhodný pri častom *Round Trippingu*, teda situácii, keď sa dokument vyberá z databáze celý, tak ako sa vložil. Druhý spôsob je vhodnejší pre situácie, keď sa databázu častejšie dotazuje pomocou dotazovacieho jazyka, keďže dotazovanie obvykle prebieha bez nutnosti parsovania dokumentu.

Dátový model ukladaných dát je potom závislý na tom, či sú dokumenty ukladané ako *intact* alebo *fine-grained*. V prvom prípade moc na dátovom modeli nezáleží, keďže je buď vrátený celý dokument alebo pri dotazovaní je nutné dokument zparsovať. V druhom prípade už je model obvykle prispôsobený dotazovaciemu jazyku. V prípade Berkeley XML DB sa jedná hlavne o XQuery pre dotazovanie a XQuery Update Facility pre úpravu dokumentov. Je taktiež možné zmeniť model na XML Infoset alebo DOM, ale v tom prípade nebude dostupná plná funkcionalita XQuery 1.0 alebo XPath 2.0.

Fyzické uloženie dát sa realizuje pomocou B-Stromov, kde id uzlov sú alokované v poradí dokumentu, ktorý je taktiež smerom iterácie nad B-Stromom. To znamená, že ak je už uzol nájdený, navigácia nad jeho detskými uzlami sa vykonáva pomocou iterácie, namiesto ďalších vyhľadávacích operácií. Schéma pre číslovanie uzlov (*numbering schema* [15]), ktorú používa Oracle Berkeley XML DB, umožňuje niektoré priame porovnania vzťahov medzi uzlami a pokúša sa o minimalizáciu potreby zostavovania uzlov pre potreby navigácie. Taktiež využíva mapovanie často sa vyskytujúcich reťazcov (názvy elementov, a podobne) na číselný tvar, ktorý umožňuje šetriť diskovým priestorom a zrýchliť prehľadávanie (porovnanie čisiel je rýchlejšie ako porovnanie reťazcov.)

Pre indexovanie dokumentov sa využívajú štrukturálne indexy, ktoré sú užitočné najmä pre navigačnú časť dotazu. Tie sú kombinované s indexami hodnôt (hodnoty elementov, atribútov, . . .) a fulltextvými indexami s podporou regulárnych výrazov.

Berkeley XML Db funguje ako knižnica, ktorá sa spúšťa priamo v aplikácii, to znamená, že nie je nutný žiaden externý databázový server. Ďalej podporuje API pre jazyky Java, C++, Perl, PHP alebo je možné komunikovať pomocou príkazového riadku.

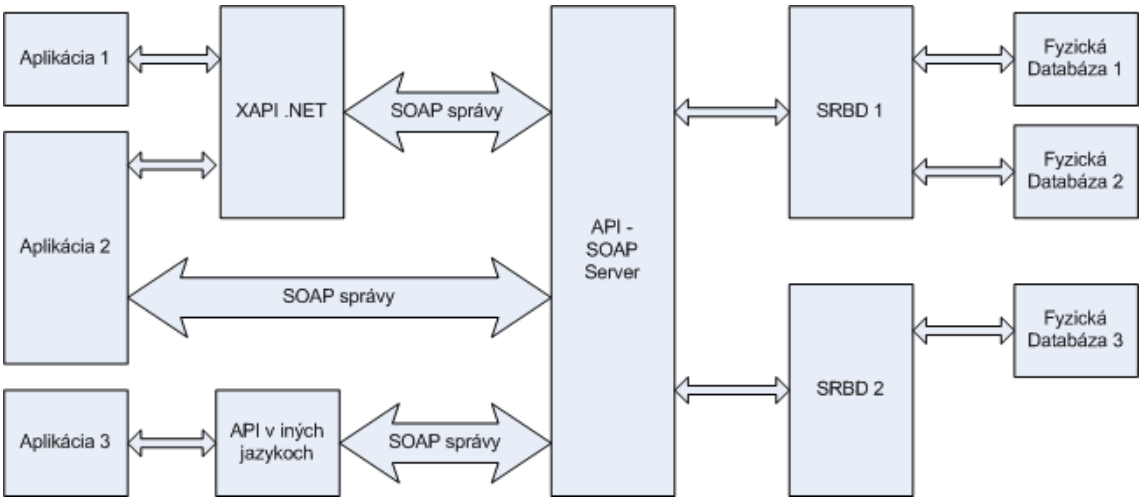
[3] Teoretické informácie o vnútornom fungovaní databáze boli prebraté zo zdroja [21]

4.3 Analýza a návrh

V následujúcej časti bude špecifikovaná architektúra, prípady užitia a statický pohľad na vyvíjané API. Usecasy sú rozdelené na 2 časti podľa používanej úrovne komunikácie : pohľad SOAP (4.3.2) a pohľad XAPI (4.3.3). V každej úrovni je uvedený use-case diagram (soap: diagram 5, XAPI: diagram 6) a popis jednotlivých prípadov užitia.

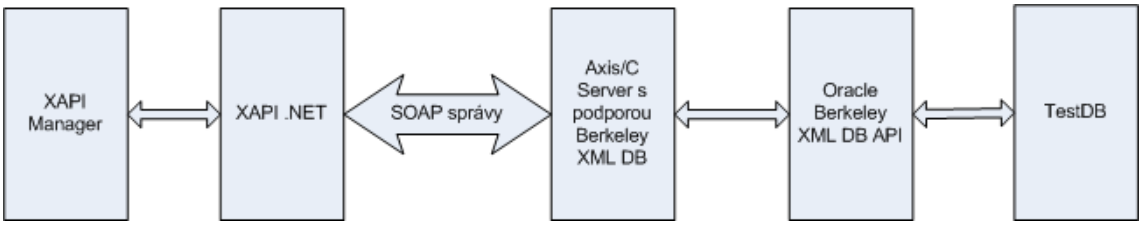
4.3.1 Architektúra riešenia

Na obrázku 3 je zobrazená navrhovaná architektúra vyvíjaného rozhrania. Na základe špecifikácie komunikácia prebieha pomocou výmeny SOAP správ, pričom XAPI rozhranie tvorí akúsi nadstavbu tohto systému. Aplikácie môžu komunikovať so serverom, buď priamo alebo pomocou spomínaného XAPI rozhrania vytvoreného ako knižnica pre .NET. Z pohľadu vývoja aplikácií je samozrejme odporúčané nadstavbové rozhranie, ktoré je komfortnejšie a bude vyvinuté podľa štandardov XmlDb.org. Zároveň táto architektúra založená na SOAP necháva možnosť jednoduchej implementácie iných typov rozhraní v prakticky ľubovoľnom jazyku, ktorý je schopný pracovať s http spojením. Špecifikácia SOAP správ je uvedená v priloženom WSDL súbore.



Obrázek 3: Architektúra riešenia API - všeobecný pohľad

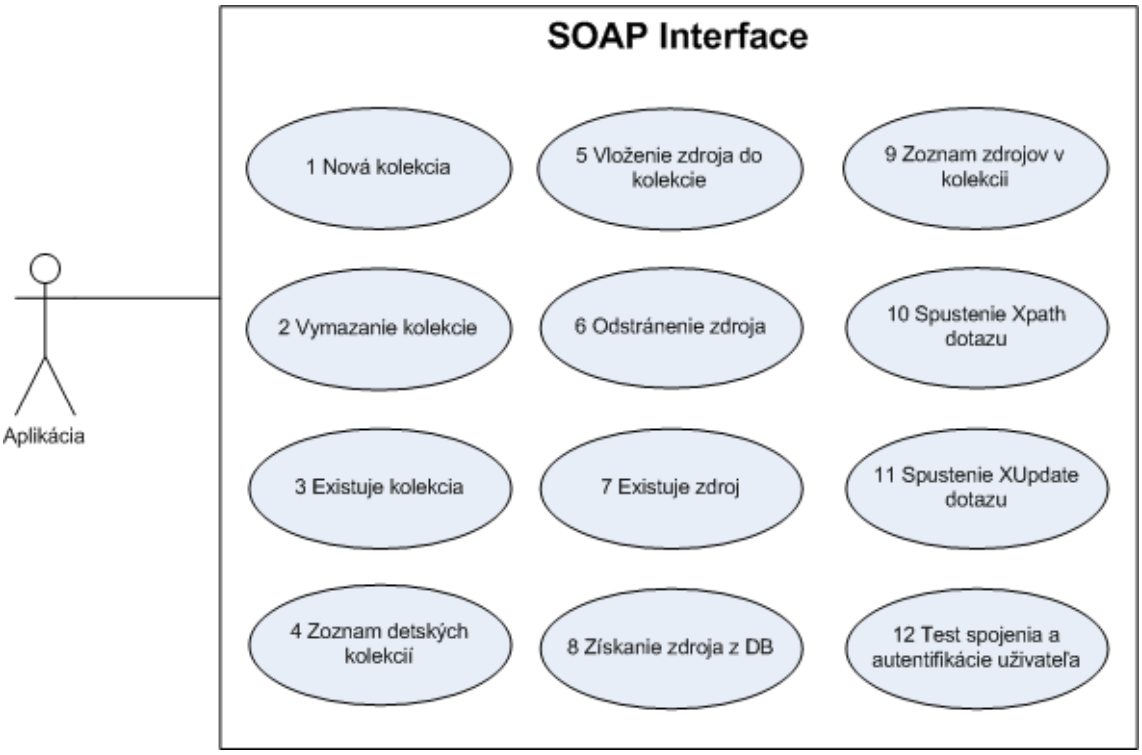
Serverová strana bude napísaná ako webová služba vo frameworku Axis2/C, pričom bude ukážkovo pripojená k SRBD Oracle Berkeley XML DB. Serverová strana bude jednoducho pripojiteľná na ľubovoľný iný SRBD. Pričom komunikácia pomocou SOAP robí server , rovnako ako klient, veľmi flexibilný voči zmenám. Je napríklad možné použiť ktorýkoľvek iný SOAP framework v kombinácii s ľubovoľnou databázou. Na serveri bude možné pracovať s viacerými rôznymi databázami (aj rôznymi typmi) rozlíšenými pomocou ich URL.



Obrázek 4: Konkrétny model vyvíjaných častí

Konkrétne vyvíjané časti systému a ich prepojenie je uvedené na obrázku 4. Ako už bolo spomínané v zadaní, server bude obsahovat' napojenie na Oracle Berkley XML DB, pričom pre potreby testovania a prezentácie bude vytvorená testovacia databáza TestDB. Pre potreby prezentácie bude taktiež vytvorená ukážková aplikácia XAPI Manager, ktorá umožní kompletnú správu databází a ich dotazovanie cez XAPI rozhranie. Aplikácia bude slúžit' najmä ako demonštrácia funkciality a ako zdroj ukážok používania XAPI v zdrojovom kóde.

4.3.2 Prípady užitia - pohľad SOAP



Obrázek 5: Usecase diagram pre SOAP klienta

Diagram 5 popisuje štruktúru prípadov užitia komunikácie priamo pomocou SOAP správ z ľubovoľného jazyka. Každý usecase reprezentuje typ správy, ktorý je možné zaslať na server nezávisle a bez špecifikácie poradia. Jedná sa o bezstavovú architektúru, podobnú architektúre REST, kde nie sú definované stavy systému. Všetky správy teda obsahujú všetky potrebné informácie (autentifikačné, názov kolekcie, ...). Tento fakt je daný tým, že použitý SOAP framework nepodporuje *sessions*, teda stavy by sa realizovali pomerne zložito.

1 Nová kolekcia Vytvorí v databázi novú kolekciu dokumentov.

- Základný tok udalostí:
  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server overí, že kolekcia s uvedeným ešte neexistuje.
  4. Server vytvorí kolekciu s daným menom a odošle aplikácii správu o výsledku operácie.
- Alternatívne toky udalostí:

  2a Uživateľské meno alebo heslo je nesprávne.

  2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

  2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

  3a Kolekcia už existuje.

  3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o existencii kolekcie s daným menom.

  3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

2 Vymazanie kolekcie Odstráni požadovanú kolekciu z databáze.

- Základný tok udalostí:
  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server overí existenciu kolekcie.
  4. Server vymaže kolekciu z databáze a odošle aplikácii správu o výsledku operácie.
- Alternatívne toky udalostí:

2a Uživateľské meno alebo heslo je nesprávne.

2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

3a Kolekcia neexistuje.

3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii kolekcie s daným menom.

3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

- Doplňujúce informácie:
  - Nie je možné vymazať *koreňovu kolekciu*.

3 Existuje kolekcia V prípade, že kolekcia v databázi existuje, vráti status 0 (OK), inak vráti príslušnú chybovú hlášku.

- Základný tok udalostí:
  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server zistí stav existencie kolekcie a odošle aplikácii správu o výsledku.
- Alternatívne toky udalostí:

2a Uživateľské meno alebo heslo je nesprávne.

2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

4 Zoznam detských kolekcií Vráti zoznam kolekcií, ktoré sú priamymi potomkami uvedenej kolekcie.

- Základný tok udalostí:
  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server overí existenciu kolekcie.
  4. Server vráti zoznam podkolekcií nájdených v adresárovom priestore kolekcie.
- Alternatívne toky udalostí:

2a Uživateľské meno alebo heslo je nesprávne.

2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

3a Kolekcia neexistuje

3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii kolekcie s daným menom.

3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

5 Vloženie zdroja do kolekcie Vloží dokument do uvedenej kolekcie.

- Základný tok udalostí:

1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou (označí začiatok posielania dokumentu : startBit=1, endBit=0).
2. Server overí autentifikačné údaje zo správy.
3. Server overí existenciu kolekcie.
4. Server overí neexistenciu zdroja v kolekcii a temp adresári (pre prípad, že niekto odosiela do kolekcie rovnaký dokument).
5. Server vygeneruje `id` pre daný zdroj a zistí veľkosť súbora, ktorú uloží do premennej `file size`.
6. Server vytvorí dočasný súbor `temp file` s názvom v tvare : názov dokumentu + `id` a vloží do neho diel dokumentu zo SOAP.
7. Server odošle správu aplikácii o priebehu operácie spolu s vygenerovaným `id` (pojistka proti zmiešaniu dokumentov rôznych klientov) a `file size` (pojistka proti duplikácii správ pri krátkom prerušení spojenia).
8. Aplikácia zostaví SOAP správu s ďalšou časťou dokumentu, pričom pridá `id` a `file size` získané zo servera. Ak sa jedná o poslednú časť dokumentu pokračuje bodom 13.
9. Aplikácia v správe nastaví startBit na 0, endBit na 0 a odošle správu na server.
10. Server vykoná rovnaké kroky ako body 2, 3.
11. Server vyhľadá `temp file` podľa mena zdroja a `id` a overí jeho veľkosť.
12. Server pridá nakoniec `temp file` časť dokumentu zo SOAP,odošle aplikácii správu s `id` a aktuálnou `file size` a pokračuje krokom 8.
13. Aplikácia v správe nastaví startBit na 0, endBit na 1 a odošle správu na server.
14. Server vykoná rovnaké kroky ako body 2, 3, 11.
15. Server pridá na koniec `temp file` časť dokumentu zo SOAP.

16. Server vloží `temp file` do kolekcie, vymaže ho z temp adresára a odošle aplikácii správu o výsledku operácie.

- Alternatívne toky udalostí:

1a Aplikácia ukladá dokument, ktorý je schopná odoslať v jednej SOAP správe.

1a1 Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou (označí začiatok posielania dokumentu : startBit=1, endBit=1).

1a2 kroky 2 - 4 sú rovnaké ako v hlavnom toku.

1a5 Server uloží dokument do kolekcie a odošle aplikácii správu o výsledku operácie, usecase týmto končí .

2a Uživateľské meno alebo heslo je nesprávne.

2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

3a Kolekcia neexistuje.

3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii kolekcie s daným menom.

3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

4a Zdroj s daným menom už v kolekcii exituje.

4a1 Server odošle aplikácií SOAP správu s chybovým hlásením o existencii zdroja s daným menom.

4a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

11a Dočasný súbor má inú veľkosť ako je uvedená v SOAP.

11a1 Server odošle aplikácií SOAP správu, ktorá je rovnaká ako v prípade úspechu so správnou veľkosť súbora.

11a2 Prípad užitia sa vráti do bodu 8.

6 Odstránenie zdroja Odstráni daný zdroj z kolekcie.

- Základný tok udalostí:

1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
2. Server overí autentifikačné údaje zo správy.
3. Server overí existenciu kolekcie.
4. Server overí existenciu zdroja.
5. Server vymaže zdroj z kolekcie a odošle aplikácii správu o výsledku operácie.

- Alternatívne toky udalostí:

  2a Užívateľské meno alebo heslo je nesprávne.

  2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

  2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

  3a Kolekcia neexistuje.

  3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii kolekcie s daným menom.

  3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

  4a Zdroj neexistuje.

  4a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii zdroja s daným menom.

  4a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

7 Existuje zdroj V prípade, že zdroj v kolekcii existuje, vráti status 0 (OK), inak vráti príslušnú chybovú hlášku.

- Základný tok udalostí:

  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server overí existenciu kolekcie.
  4. Server zistí stav existencie zdroja a odošle aplikácii správu o výsledku.

- Alternatívne toky udalostí:

  2a Užívateľské meno alebo heslo je nesprávne.

  2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

  2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

  3a Kolekcia neexistuje.

  3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii kolekcie s daným menom.

  3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

8 Získanie zdroja z DB Vráti zdroj s daným meno z kolekcie.

- Základný tok udalostí:
  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server overí existenciu kolekcie.
  4. Server overí existenciu zdroja.
  5. Server odošle aplikácii dokument uložený pod daným menom.
- Alternatívne toky udalostí:

  2a Uživateľské meno alebo heslo je nesprávne.

  2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

  2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

  3a Kolekcia neexistuje.

  3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii kolekcie s daným menom.

  3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

  4a Zdroj neexistuje.

  4a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii zdroja s daným menom.

  4a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

9 Zoznam zdrojov v kolekcii Vráti zoznam zdrojov uložených v danej kolekcii.

- Základný tok udalostí:
  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server overí existenciu kolekcie.
  4. Server odošle aplikácii zoznam mien zdrojov uložených v kolekcii.
- Alternatívne toky udalostí:

  2a Uživateľské meno alebo heslo je nesprávne.

  2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

  2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

3a Kolekcia neexistuje.

3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii kolekcie s daným menom.

3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

10 Spustenie XPath dotazu Spustí nad danou kolekciou XPath dotaz a vráti výsledky dotazu.

- Základný tok udalostí:

  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server overí existenciu kolekcie.
  4. Server spustí XPath dotaz nad kolekciou a odošle aplikácii zoznam výsledkov v tvare definovanom vo WSDL dokumente.

- Alternatívne toky udalostí:

2a Uživateľské meno alebo heslo je nesprávne.

2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

3a Kolekcia neexistuje.

3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii kolekcie s daným menom.

3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

11 Spustenie XUpdate dotazu Spustí nad danou kolekciou XUpdate dotaz.

- Základný tok udalostí:

  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server overí existenciu kolekcie.
  4. Server spustí XUpdate dotaz nad kolekciou a odošle aplikácii oznámenie o výsledku operácie.

- Alternatívne toky udalostí:

2a Uživateľské meno alebo heslo je nesprávne.

2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

3a Kolekcia neexistuje.

3a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neexistencii kolekcie s daným menom.

3a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

12 Test spojenia a autentifikácie uživateľa Otestuje existenciu databáze na serveri a overí uživateľské údaje.

- Základný tok udalostí:

  1. Aplikácia zostaví SOAP správu v zodpovedajúcom tvare a odošle ju na adresu servera s požadovanou databázou.
  2. Server overí autentifikačné údaje zo správy.
  3. Server odošle aplikácii oznámenie o úspešnom prihlásení.

- Alternatívne toky udalostí:

  1a Aplikácia nedostane žiadnu odpoveď na správu.

  1a1 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

  2a Uživateľské meno alebo heslo je nesprávne.

  2a1 Server odošle aplikácií SOAP správu s chybovým hlásením o neúspešnom prihlásení.

  2a2 Aplikácia sa vráti do bodu 1 alebo prípad užitia končí.

4.3.3 Prípady užitia - pohľad XAPI

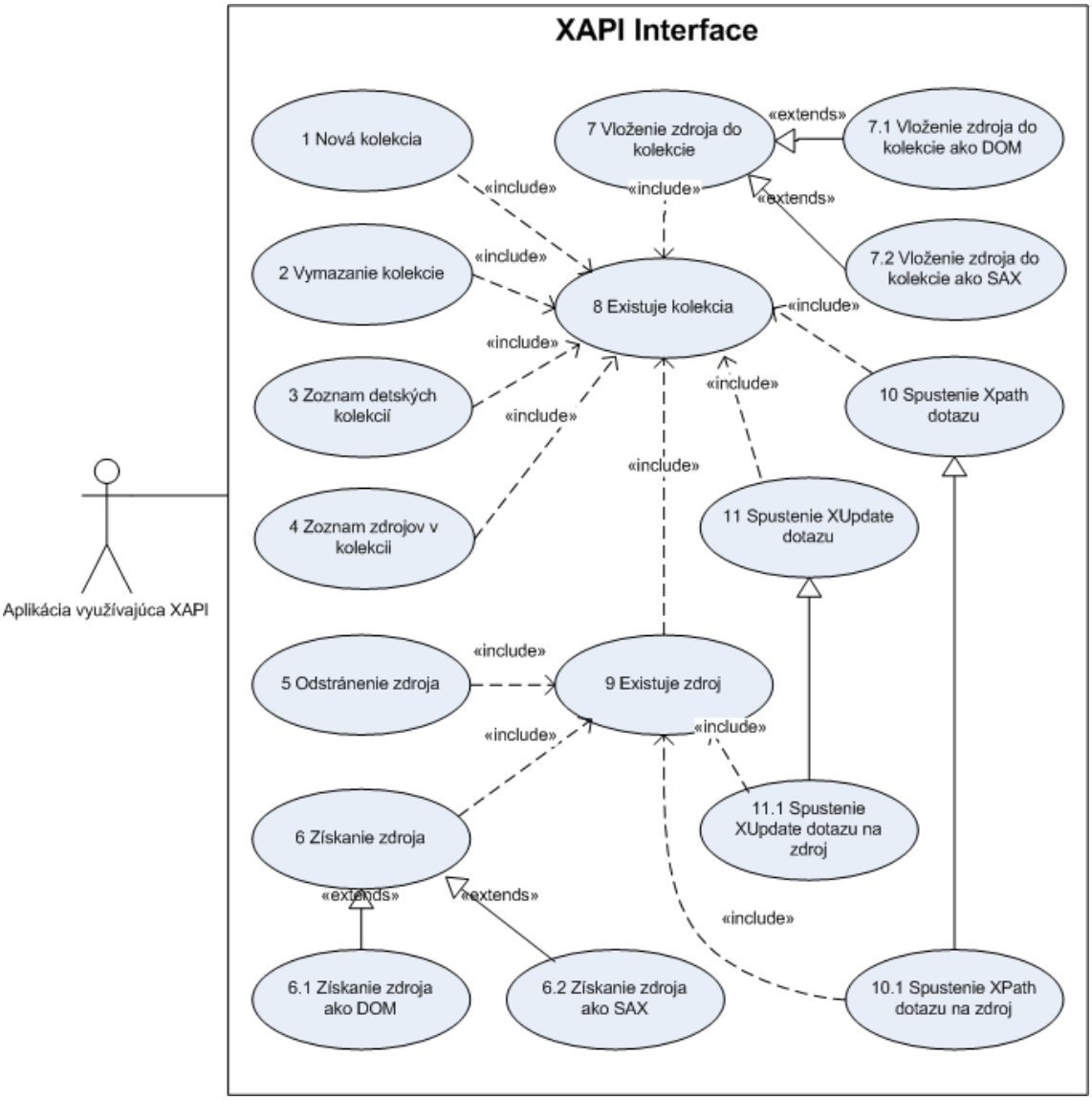
Diagram 6 popisuje hierarchiu prípadov užitia z pohľadu rozhrania XAPI. Keďže implementácia tohto interface je postavená na zasielaní SOAP správ, prípady užitia tejto vrstvy sa kvôli obmedzení duplicity priamo odkazujú na prípady užitia SOAP vrstvy. Odkazované prípady užitia zo SOAP pohľadu budú označené ako: `SOAP číslo názovUC`.

1 Nová kolekcia Vytvorí novú kolekciu v databázi.

- Predpoklady:

  – Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.

- Základný tok udalostí:



Obrázek 6: Usecase diagram pre XAPI klienta

1. Aplikácia zadá požiadavku pre získanie rodičovskej kolekcie na základe jej mena.
2. XAPI overí existenciu kolekcie pomocou `8 Existuje kolekcia` a v prípade existencie vráti reprezentáciu kolekcie `col`.
3. Aplikácia získa z `col` službu pre správu kolekcií `service`.
4. Aplikácia zavolá na `service` metódu pre vytvorenie kolekcie s menom novej kolekcie.

5. XAPI vytvorí novú kolekciu pomocou `SOAP 1 Nová kolekcia` a vráti aplikácii inštanciu novej kolekcie.

- Doplňujúce informácie:
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

2 Vymazanie kolekcie Vymaže existujúcu kolekciu z databáze.

- Predpoklady:
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.
  - Koreňovú kolekciu nie je možné odstrániť.
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie rodičovskej kolekcie na základe jej mena.
  2. XAPI overí existenciu kolekcie pomocou `8 Existuje kolekcia` a v prípade existencie vráti reprezentáciu kolekcie `col`.
  3. Aplikácia získa z `col` službu pre správu kolekcií `service`.
  4. Aplikácia zavolá na `service` metódu pre odstránenie požadovanej kolekcie.
  5. XAPI vytvorí odstráni kolekciu pomocou `SOAP 2 Vymazanie kolekcie`.
- Doplňujúce informácie:
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

3 Zoznam detských kolekcií Vráti zoznam názvov detských kolekcií danej kolekcie.

- Predpoklady:
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie rodičovskej kolekcie na základe jej mena.
  2. XAPI overí existenciu kolekcie pomocou `8 Existuje kolekcia` a v prípade existencie vráti reprezentáciu kolekcie `col`.
  3. Aplikácia zavolá na `col` metódu pre získanie zoznamu.

4. XAPI vráti aplikácii zoznam názvov podkolekcií získaný pomocou `SOAP 4 Zoznam detských kolekcií`.

- Doplňujúce informácie:
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

4 Zoznam zdrojov v kolekcii Vráti zoznam názvov detských kolekcií danej kolekcie.

- Predpoklady:
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie rodičovskej kolekcie na základe jej mena.
  2. XAPI overí existenciu kolekcie pomocou `8 Existuje kolekcia` a v prípade existencie vráti reprezentáciu kolekcie `col`.
  3. Aplikácia zavolá na `col` metódu pre získanie zoznamu.
  4. XAPI vráti aplikácii zoznam zdrojov kolekcie získaný pomocou `SOAP 9 Zoznam zdrojov v kolekcii`.
- Doplňujúce informácie:
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

5 Odstránenie zdroja Odstráni zdroj z danej kolekcie.

- Predpoklady:
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie rodičovskej kolekcie na základe jej mena.
  2. XAPI overí existenciu zdroja pomocou `9 Existuje zdroj` a v prípade existencie vráti reprezentáciu jeho kolekcie `col`.
  3. Aplikácia zavolá na `col` metódu pre odstránenie zdroja.
  4. XAPI odstráni daný zdroj pomocou `SOAP 6 Odstránenie zdroja`.

- Doplňujúce informácie:
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

6 Získanie zdroja Vráti zdroj uložený v kolekcie.

- Predpoklady:
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie rodičovskej kolekcie na základe jej mena.
  2. XAPI overí existenciu zdroja pomocou `9 Existuje zdroj` a v prípade existencie vráti reprezentáciu jeho kolekcie `col`.
  3. Aplikácia zavolá na `col` metódu pre získanie zdroja.
  4. XAPI vráti aplikácii reprezentáciu zdroja pomocou `SOAP 8 Získanie zdroja z DB`.
- Alternatívne toky udalostí:

  4a Aplikácia požaduje vrátiť zdroj ako DOM objekt.

  4a1 XAPI vráti aplikácii reprezentáciu zdroja pomocou `SOAP 8 Získanie zdroja z DB`.

  4a2 Aplikácia zadá požiadavku pre vrátenie zdroja ako DOM

  4a3 XAPI vytvorí nový DOM objekt a ako jeho obsah nastaví text zdroja z DB.

  4a4 XAPI vráti aplikácii vytvorený DOM objekt.

  4b Aplikácia požaduje spracovať zdroj pomocou SAX.

  4b1 XAPI vráti aplikácii reprezentáciu zdroja pomocou `SOAP 8 Získanie zdroja z DB`.

  4b2 Aplikácia si vytvorí SAX handler, podľa ktorého sa bude zdroj spracovávať.

  4b3 Aplikácia zadá požiadavku pre spracovanie zdroja ako SAX spolu s vytvoreným handlerom.

  4b4 XAPI spustí spracovanie pomocou SAX podľa dodaného handlera.
- Doplňujúce informácie:
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

7 Vloženie zdroja do kolekcie Uloží zdroj do kolekcie.

- Predpoklady:
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie rodičovskej kolekcie na základe jej mena.
  2. XAPI overí existenciu kolekcie pomocou `8 Existuje kolekcia` a v prípade existencie jej vráti reprezentáciu `col`.
  3. Aplikácia zavolá na `col` metódu pre vytvorenie, buď zadá jedinečné meno zdroja alebo ho nechá vygenerovať.
  4. Aplikácia priradí zdroju jeho obsah a podá požiadavku na jeho vloženie do DB.
  5. XAPI uloží reprezentáciu zdroja do kolekcie pomocou `SOAP 5 Vloženie zdroja do kolekcie`.
- Alternatívne toky udalostí:

  4a Aplikácia priradí obsah ako DOM objekt.

  4a1 Aplikácia zadá požiadavku nastavenie obsahu zdroja ako DOM, spolu s obsahom v DOM.

  4a2 XAPI transformuje DOM objekt na textovú reprezentáciu.

  4a3 Aplikácia zadá požiadavku pre vloženie zdroja do DB. Pokračuje sa bodom 5.

  4b Aplikácia priradí obsah pomocou SAX.

  4b1 Aplikácia zadá požiadavku nastavenie obsahu zdroja ako SAX.

  4b2 XAPI vráti handler pre ukladanie obsahu do pomocného súbora `temp`.

  4b3 Aplikácia spustí SAX prechádzanie.

  4b4 Handler uloží dokument do `temp`.

  4b5 Aplikácia zadá požiadavku pre vloženie zdroja do DB. Pokračuje sa bodom 5.
- Doplňujúce informácie:
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

8 Existuje kolekcia V prípade existencie kolekcie vráti `true`, inak `false`.

- Predpoklady:
  - Špecifikácia databáze je zaregistrovaná v `DatabaseManager` triede.
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie kolekcie na `DatabaseManager` spolu s uri v požadovanom tvare.
  2. XAPI overí existenciu kolekcie pomocou `SOAP 3 Existuje kolekcia`, ak existuje vráti hodnotu `true`, inak `false`.
- Doplňujúce informácie:
  - Požadovaný tvar uri : xmldb:typ databáze://adresa servera/názov databáze/adresa v stromovej štruktúre kolekcií (ich názvy oddelené /)/názov kolekcie, napríklad : `xmldb:berkeley://localhost:9090/axis2/services/XmlDBServer/test`, kde XmlDBServer je názov databáze a test je názov kolekcie.
  - V prípade dotazovania *root* kolekcie sa názov kolekcie neuvádza, to znamená, že uri končí názvom databáze.
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

9 Existuje zdroj V prípade existencie zdroja v kolekcii vráti `true`, inak `false`.

- Predpoklady:
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie kolekcie na základe mena .
  2. XAPI overí existenciu kolekcie pomocou `8 Existuje kolekcia` a v prípade existencie vráti reprezentáciu kolekcie `col`.
  3. XAPI overí existenciu zdroja v kolekcii pomocou `SOAP 7 Existuje zdroj`, ak existuje vráti hodnotu `true`, inak `false`.
- Doplňujúce informácie:
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

10 Spustenie dotazu XPath Spustí dotaz jazyka XPath na kolekciu alebo priamo na zdroj a vráti výsledky.

- Predpoklady:
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie kolekcie na základe jej mena.
  2. XAPI overí existenciu kolekcie pomocou `8 Existuje kolekcia` a v prípade existencie vráti jej reprezentáciu `col`.
  3. Aplikácia získa z `col` službu pre prácu s XPath dotazmi a spustí na ňu požadovaný dotaz.
  4. XAPI odošle dotaz na server pomocou `SOAP 10 Spustenie XPath dotazu` a vráti jeho výsledky aplikácii, ako množinu zdrojov.
- Alternatívne toky udalostí:

  4a Aplikácia spúšťa dotaz priamo na zdroj

  4a1 XAPI upraví dotaz pre spustenie na konkrétny zdroj.

  4a2 Usecase pokračuje bodom 4.
- Doplňujúce informácie:
  - V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

11 Spustenie dotazu XUpdate Spustí dotaz jazyka XUpdate na kolekciu alebo priamo na zdroj a vráti výsledky.

- Predpoklady:
  - Predpokladá sa, že koreňová kolekcia existuje vždy, je vytvorená pri vytvorení databáze.
- Základný tok udalostí:
  1. Aplikácia zadá požiadavku pre získanie kolekcie na základe jej mena.
  2. XAPI overí existenciu kolekcie pomocou `8 Existuje kolekcia` a v prípade existencie vráti jej reprezentáciu `col`.
  3. Aplikácia získa z `col` službu pre prácu s XUpdate dotazmi a spustí na ňu požadovaný dotaz.
  4. XAPI odošle dotaz na server pomocou `SOAP 11 Spustenie XUpdate dotazu` a vráti jeho výsledky aplikácii, ako množinu zdrojov.

- Alternatívne toky udalostí:

  5a Aplikácia spúšťa dotaz priamo na zdroj

  5a1 XAPI upraví dotaz pre spustenie na konkrétny zdroj.

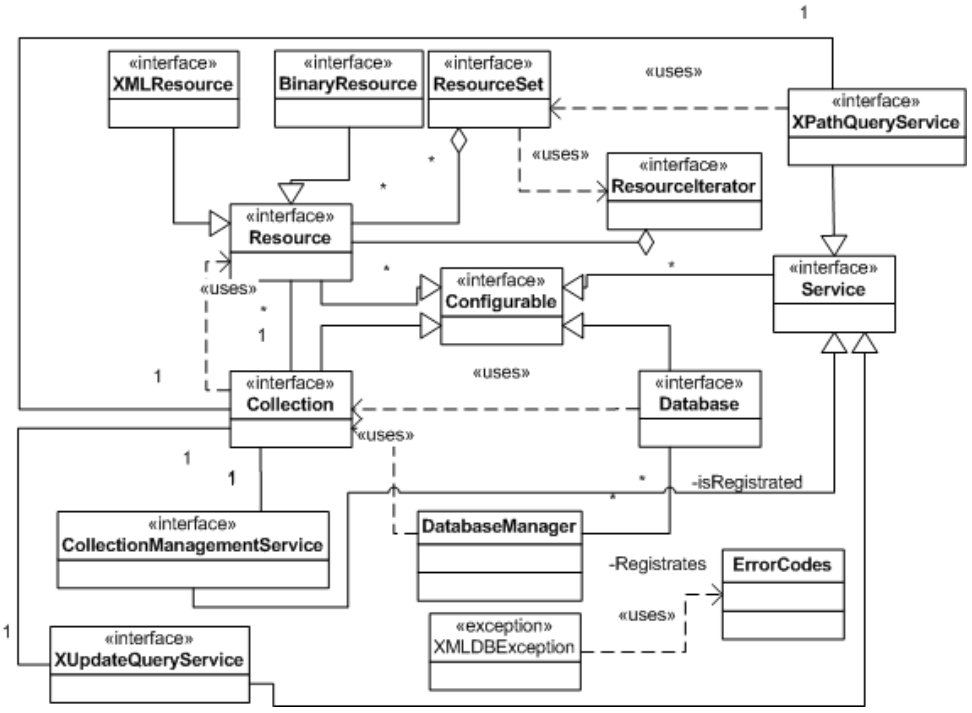
  5a2 Usecase pokračuje bodom 5.

- Doplňujúce informácie:

  – V prípade neúspešného vykonania niektorého kroku vyhodí XAPI vlastnú výnimku s číslo a popisom problému.

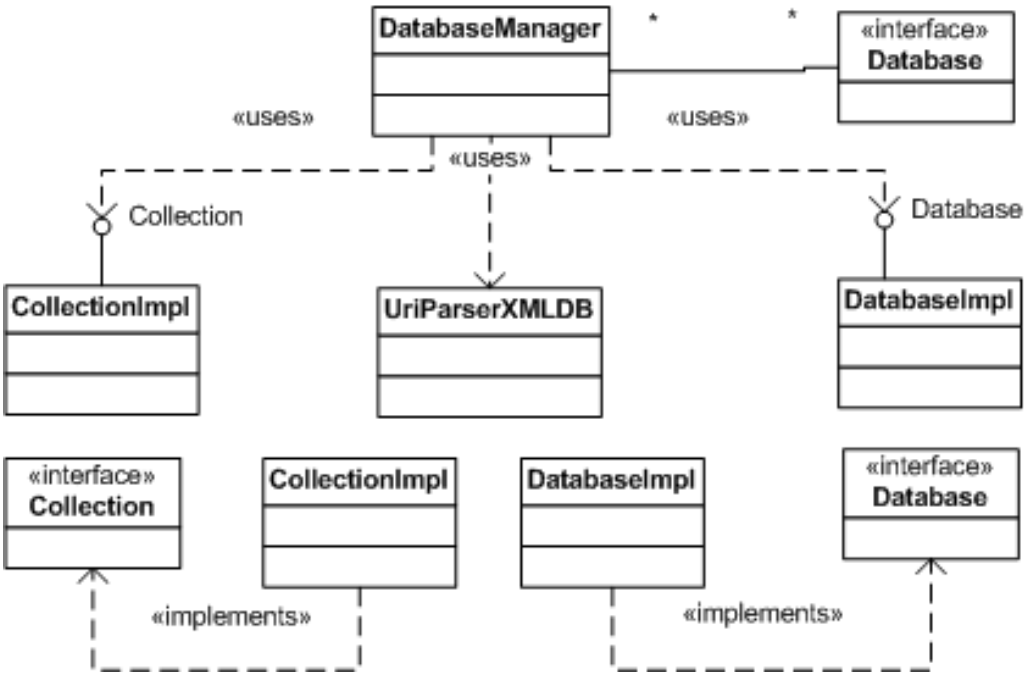
4.3.4 Statický pohľad na XAPI

Statickú štruktúru zachytáva triedny diagram, ktorý je pre väčšiu prehľadnosť rozdelený na niekoľko dielov. Prvá časť (na obrázku 7) zobrazuje samotnú štruktúru rozhrania XAPI, teda sa jedná o prehľad rozhraní, ktoré budú priamo prístupné aplikáciám. Základom je trieda DatabaseManager, ktorá slúži na registráciu jednotlivých databázových typov a na základe uri je schopná vrátiť požadovanú kolekciu požadovaného typu databáze. Klientská aplikácia následne už využíva hlavne objekt typu Collection, cez ktorý je možné narábať s jednotlivými zdrojmi, spravovať kolekcie a dotazovať dáta. Podrobný popis rozhrania, vrátane metód a parametrov je možné nájsť na stránke iniciatívy XMLDB.org.



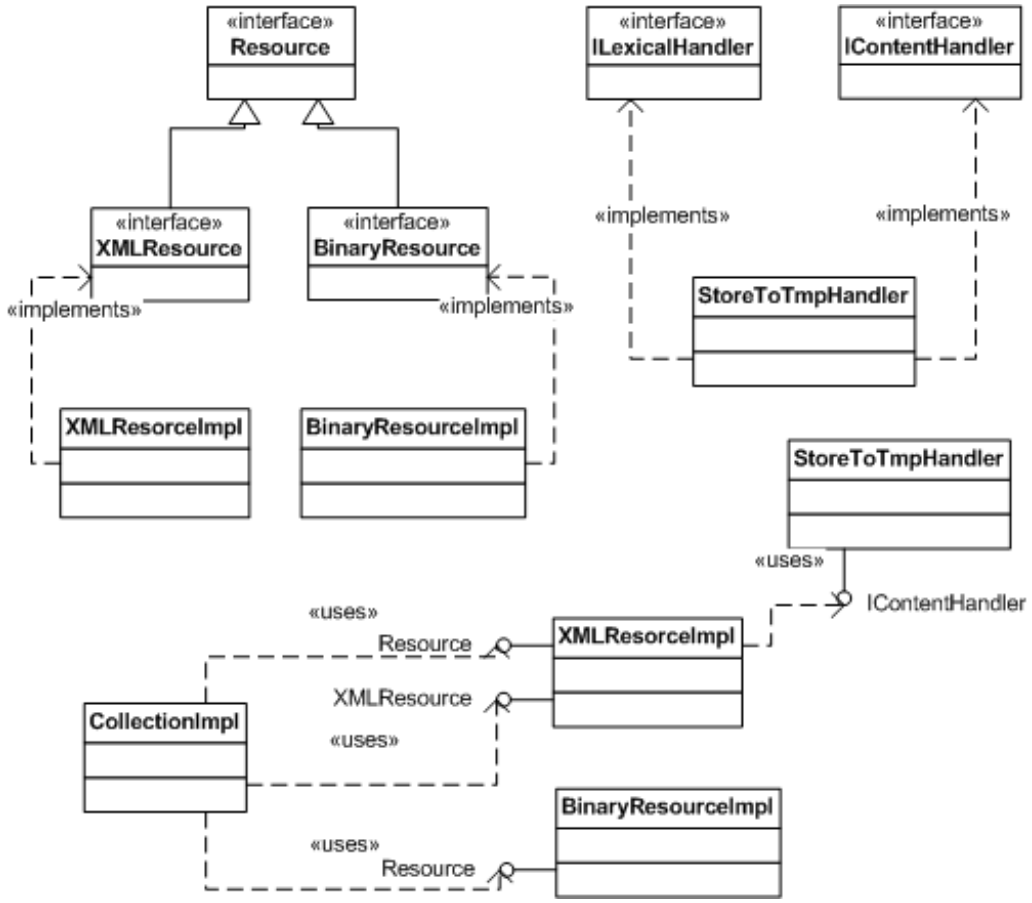
Obrázek 7: Triedny diagram zobrazujúci štruktúru XAPI

Ako už bolo spomínané, základom modelu XAPI je trieda `DatabaseManager`, ktorá registruje všetky typy databází, s ktorými je možné komunikovať pomocou tohto interface. Pre registráciu v managerovi je nutné, aby trieda implementovala rozhranie `Database`, cez ktoré s ním komunikuje. V našom prípade sa jedná o triedu `DatabaseImpl`, ktorá je aktuálne prispôsobená pre komunikáciu s databázovým systémom Oracle XML DB. Pre získanie kolekcie z databáze je využíva metóda `getCollection`, ktorá na základe zadaného uri zistí (pomocou triedy `UriParserXMLDB`), do ktorej databáze patrí a vráti rozhranie `Collection`, konkrétne implementované v triede `CollectionImpl`. Toto rozhranie sa následne využíva pre prácu s dokumentami a službami kolekcie. Statický model situácie je uvedený na obrázku 8.



Obrázek 8: Triedny diagram popisujúci správu databází a získanie kolekcie

Na diagrame 9 je uvedený statický model práce s dokumentami. Dokumenty sa vo všeobecnosti delia na 2 typy : XML dokument a binárne dáta. Narába sa s nimi pomocou rozhraní `XMLResource` a `BinaryResource`. Rozhranie podporuje obidva typy, ale v konkrétnej implementácii pripojenej k databáze Oracle binárne dokumenty podporované nie sú. Operácie, ako získanie dokumentu alebo uloženie do databáze, sú vykonávané rozhraním `Collection`. S XML dokumentami je možné pracovať 2 spôsobmi: ako s DOM objektom alebo pomocou SAX handlera. Pre spracovanie pomocou SAX sa aplikácii vráti inštancia `StoreToTmpHandler` implementujúca SAX rozhrania IContentHandler a ILexicalHandler. Tento handler uloží dokument do dočasného súbora, s ktorým sa neskôr pracuje.

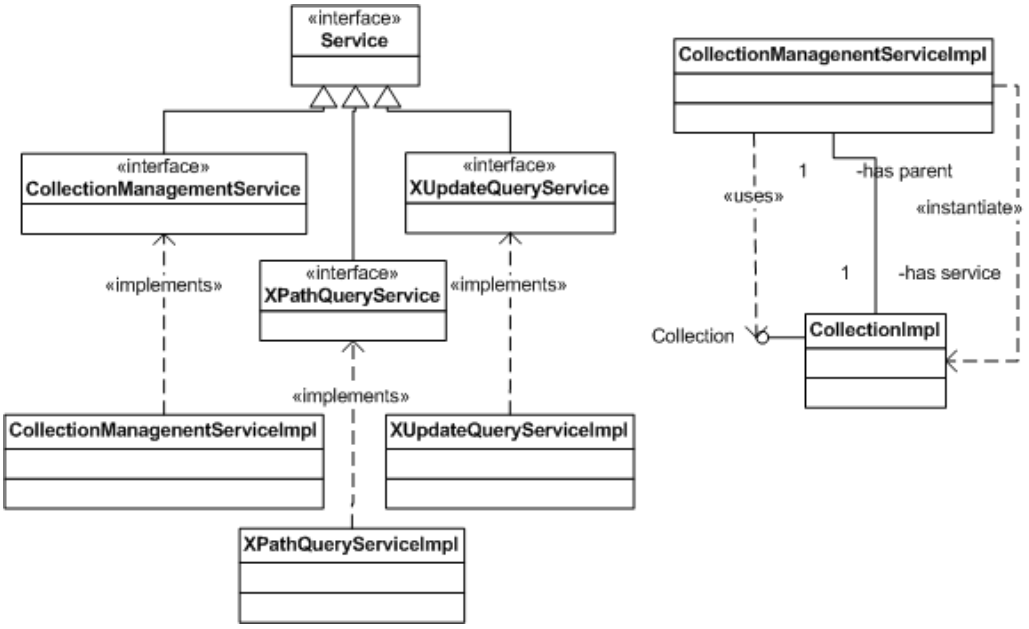


Obrázek 9: Statický model práce s dokumentami

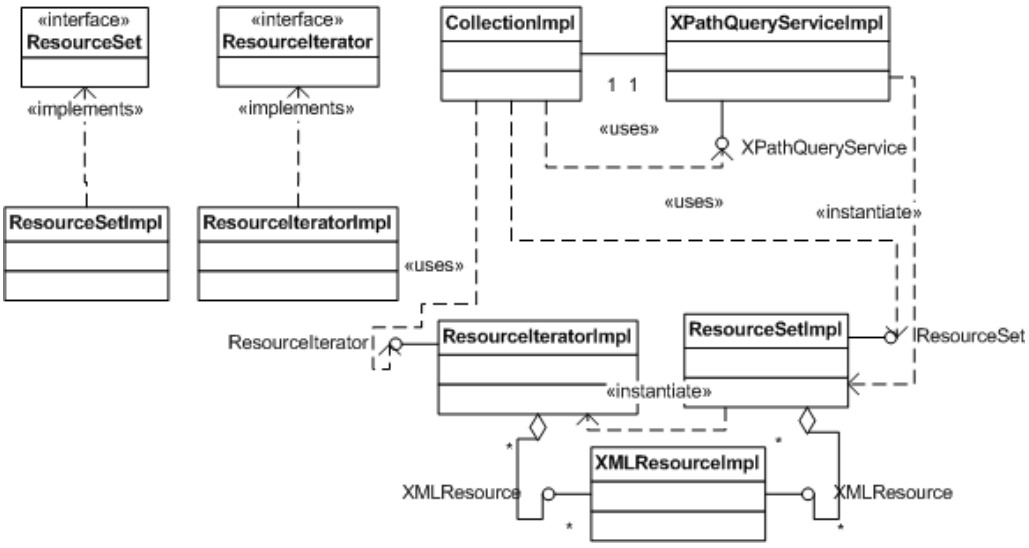
Dôležitou súčasťou XAPI je takzvaný systém služieb - *services* (definovaný v diagrame 10). Služby sa starajú o prídavnú funkcionalitu, a keďže nie je problém v prípade nutnosti vytvoriť ďalšie služby pre ľubovoľné prídavné funkcie, umožňujú XAPI veľkú mieru modularity. V aktuálnej implementácii sa zatiaľ počíta s 3 službami:

- CollectionManagementService - pre správu kolekcií, definovaná v diagrame 10
- XPathQueryService - pre dotazovanie v jazyku XPath
- XUpdateQueryService - pre dotazovanie v jazyku XUpdate

Ďalšou funkcionálnou časťou rozhrania je dotazovanie kolekcie pomocou jazyka XPath (diagram 11). Narozdiel od XUpdate tento jazyk vracia nejaký výsledok, z toho dôvodu je mu venovaná samostatná časť. Dotazovanie prebieha pomocou služby XPathQueryService. Výsledok takého dotazu je dostupný pomocou rozhrania `ResourceSet`, prípadne `ResourceIterator`. V oboch prípadoch je možné výsledky jednoducho prechádzať a prípadne s nimi pracovať ako so samostatnými zdrojmi.



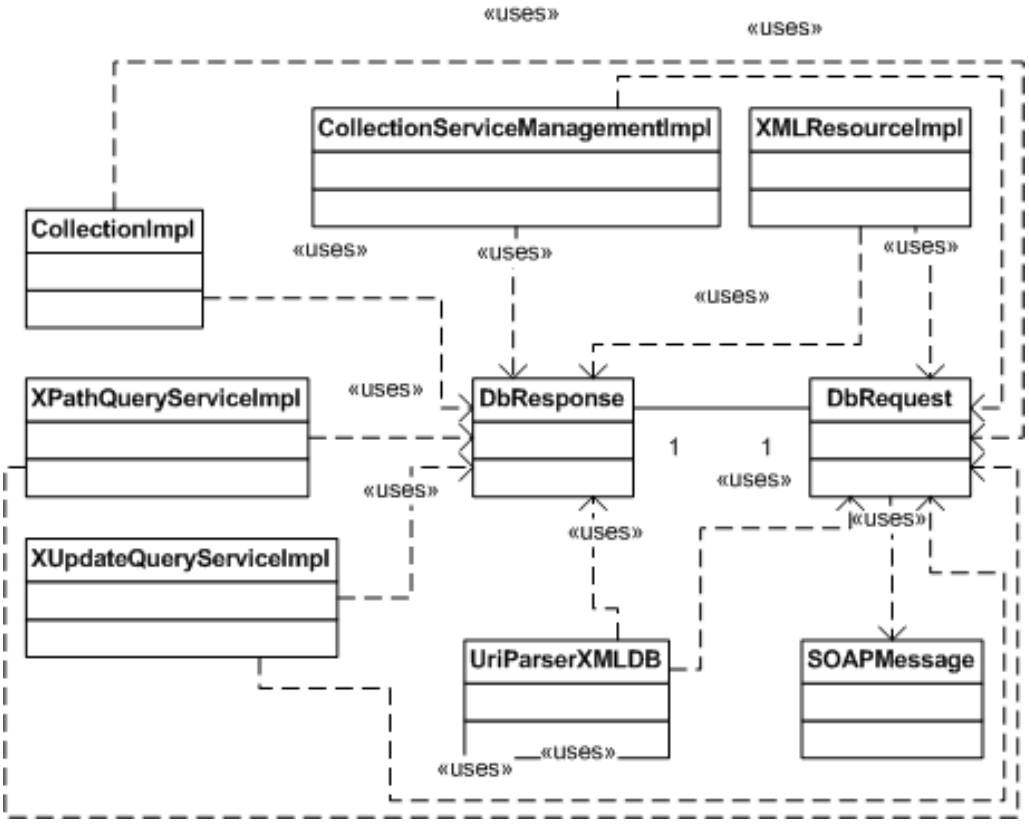
Obrázek 10: Model zobrazujúci systém služieb a prácu s kolekciami



Obrázek 11: Model zobrazujúci dotazovanie pomocou XPath

Posledný triedny diagram 12 modeluje samotnú komunikáciu so serverom databáze, resp. rozhrania. Keďže systém komunikácie je navrhnutý ako zasielanie SOAP správ, triedy XAPI budú zasielať svoje požiadavky týmto spôsobom. Odosielanie je zaobalené do triedy `DbRequest`, ktorá zjednodušuje prácu so SOAP. Samotné správy sú vytvárané

a odosielané v triede `SOAPMessage`. Po získaní výsledku je vrátený vo forme triedy `DbResponse`.



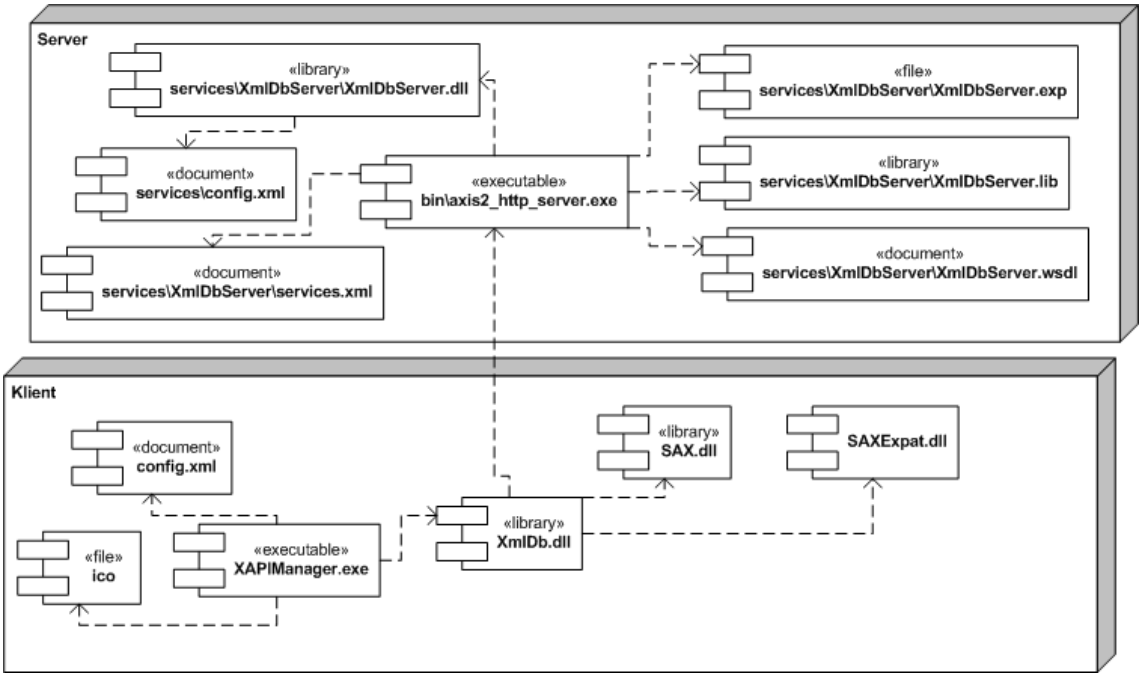
Obrázek 12: Komunikácia pomocou SOAP v XAPI

4.4 Implementácia

4.4.1 Diagram nasadenia

Na obrázku 13 je zobrazený diagram nasadenia, ktorý popisuje závislosti medzi jednotlivými časťami systému, knižnicami a podpornými dokumentami nutnými pre správny chod rozhrania a ukážkovej aplikácie. Základom serverovej časti je XmlDbServer.dll, spolu s podporným .lib a .exp súbormi rovnakého mena. Zložka s týmito súbormi je uložená v adresári *services*. Názov zložky (v našom prípade *XmlDbServer*) je taktiež názvom databáze na serveri, teda pre testovanie bude v adresári *services* zložka *TestDB*. Ďalšou povinnou súčasťou sú súbory XmlDbServer.wsdl a services.xml. Wsdl súbor obsahuje popis a štruktúru všetkých podporovaných SOAP správ, ktoré je schopný spracovať. Súbor slúži najmä pre vývojárov, ktorí sa rozhodnú vyvíjať svoje vlastné klientské rozhranie alebo komunikovať s databázou priamo. Súbor services.xml je zasa povinnou súčasťou

webovej služby v Axis2/C, kde popisuje aké operácie služba podporuje. Oba súbory úzko spolupracujú a popisujú veľa spoločných častí, takže je nutné ich pri vývoji synchronizovať. Pri používaní ich stačí iba prekopírovať, prípade ešte premenovať. Špecifickým prípadom je konfiguračný súbor config.xml, ktorý je spoločný pre všetky používané databáze. Config je xml súbor s 2 povinnými elementami: */config/storage_folder* - udáva cestu, kam sa budú ukladať dáta databází a */config/os_type*, ktorý špecifikuje typ operačného systému. Samotný server sa spúšťa pomocu *axis_http_server.exe* alebo je integrovaný do servera Apache.



Obrázek 13: Diagram nasadenia vyvíjaného rozhrania

Na strane klienta je kľúčovým súborom XmlDb.dll, táto knižnica obsahuje kompletne implementované rozhranie XAPI a priamo komunikuje so serverom. Pre jej správne fungovanie je nutné mať v adresári ešte knižnicu SAX.dll, SAXExpat.dll a ostatné dll knižnice dodávané v balíku pre podporu SAX parsovania. Knižnice je možné získať zo stránok projektu SAX pre .NET prostredie. Ukážková aplikácia XAPI Manager sa spúšťa prostredníctvom rovnomenného exe súbora. Aplikácia potrebuje okrem samotnej knižnice XmlDB a jej závislostí ešte súbor config.xml a zložku s ikonami ico (ikony sú využívané priamo v aplikácii, teda sú nevyhnutné pre správne fungovanie). Konfiguračný súbor config.xml špecifikuje konfiguráciu rozhrania XAPI (nastavenia automatickej obnovy spojenia, nastavenia poolov, . . .), zoznam databází a ich údajov pre pripojenie a ďalšie nastavenia aplikácie (napríklad: zobrazovanie zdrojov v kolekciách alebo aktivity debugovacieho módu).

4.4.2 Problémy pri implementácii

4.4.2.1 Dokumentácia Jedným z prvých problémov, ktorý sa vyskytol pri implementácii rozhrania, bola nedostatočná, prípadne slabá dokumentácia. Tento problém sa vyskytuje najmä u open-source projektov, kde je veľa nezávislých vývojárov. Konkrétne sa tento problém vyskytol pri implementácii serverovej časti v prostredí Axis2/C a pri používaní knižnice SAX. Projekt Axis2 je rozdelený na 2 základné skupiny podľa jazyka, pre ktorý je vyvíjaný: C a Java. Dokumentácia pre Java platformu je na veľmi dobrej úrovni, kde nechýba podrobný popis a konkrétne ukážky použitia. Bohužiaľ v prípade jazyka C je to oveľa horšie, chýba najmä ucelený a komplexný prehľad funkcií, ich popis a ukážky použitia. V praxi preto bolo nutné prechádzať jednotlivé *h súbory* a podľa komentárov a názvov funkcií hľadať vyhovujúcu funkciu, či dokonca zisťovať, či vôbec existuje. Podobný prípad nastal aj v spomínanej knižnici SAX, i keď už v nie takom rozsahu. SAX sa prioritne vyvíja pre platformu Java, preto pre .Net verziu bola dokumentácia o dosť chudobnejšia. Avšak v tomto prípade sa dala v rozumnej miere využiť aj Java dokumentácia, keďže obe knižnice sú do značnej miery podobné.

4.4.2.2 Emulácia stromovej štruktúry kolekcií Keďže databáza Oracle Berkeley XML DB priamo nepodporuje stromový systém kolekcií, musela byť táto vlastnosť emulovaná. V praxi sa využíva systém zložiek súborového systému, kde názov zložky je názov podkolekcie a v zložke je vždy práve jeden súbor _col.dbxml, ktorý obsahuje danú kolekciu. Pričom každá zložka môže obsahovať ľubovoľné množstvo podadresárov (podkolekcií).

4.4.2.3 Kódovanie textu a XML entity Udržanie konzistencie kódovania textu a tým pádom aj celých XML dokumentov sa ukázalo ako veľmi nepríjemný problém. Hlavným dôvodom boli rôzne východzie nastavenia pri spracovaní vo vnútri .NET (dátový typ string a DOM reprezentácia XML dokumentov), pri prenose pomocou SOAP a pri spracovaní na serveri a nakoniec aj pri uložení do databáze. Nakoniec sa, ale podarilo kódovanie zosynchronizovať na kódovanie UTF-8. Vedľajším efektom tohto problému bolo aj vytváranie takzvaných preddefinovaných XML entít pri prenose. Napríklad znak < sa preložil na <. Celkovo ich je v XML predifinovaných 5 : < > & " a '. Pre mnohé databáze tento preklad síce neznamená problém, ale práve v prípade Oracle Berkeley XML DB sa prejavil tým, že nebolo možné takéto dokumenty ukladať. Problém sa ani po viecerých pokusoch nepodarilo jednoducho odstrániť. Preto musel byť použitý pomerne neohrabaný spôsob, ktorý spočíval v prepisovaní týchto znakov v správe po jej prijatí.

4.4.2.4 SAX knižnica Ako už bolo spomínané SAX knižnica je primárne vyvíjaná pre jazyk Java a pre .NET sa jedná skôr o okrajovú záležitosť. Dôvodom prečo bola vôbec použitá (.NET obsahuje aj vlastné riešenie pre parsovanie pomocou udalostí) je, že sa to explicitne vyžaduje v dokumentácii XAPI (ktoré bolo taktiež vyvíjané primárne pre Javu, takže tam to problém nebol). Napriek mnohým častiam, ktoré sú prakticky identické, je možné nájsť viacero rozdielov. Tieto rozdiely v3ak v niektorých prípadoch zname-

nali pomerne veľké funkcionálne problémy. Napríklad rozdiely v štruktúre handlerov (ContentHandler a LexicalHandler), ale najmä pri samotnom spúšťaní parsera. V Jave napríklad nie je problémom pomocou parsera prechádzať reťazce znakov, pokiaľ sa použije trieda StringReader, bohužiaľ toto v .NET prostredí nie je možné. Respektíve sa to nedarí pomocou podobnej triedy v tomto prostredí, a keďže dokumentácia k tej časti funkcionality v .NET neexistuje, bolo nutné pred parsovaním vytvoriť dočasný súbor a prechádzať ten. Čo síce problém vyriešilo, ale znamenalo zavedenie zbytočných krokov a zníženie výkonu.

4.4.2.5 Rozdiely medzi Javou a C# Rozhranie XAPI bolo primárne vyvíjané pre jazyk Java, to by samo o sebe pre vývoj v .NET problém nebol, keďže jazyky sú to prakticky totožné. Ale predsa len je možné naraziť na malé rozdiely, ktoré dokážu ovplyniť štruktúru kódu. Jedným z týchto detailov je napríklad nemožnosť definovať v rozhraní (typ Interface) statickú konštantu a jej hodnotu. Táto vlastnosť sa v XAPI hojne využíva, pretože v Jave to možné je. Rešení bolo teda vytvoriť triedu so statickými konštantami, ktoré sa vyskytujú v celom programe (konkrétne sa jedná o triedu *XmlDBTypes*). Iným riešením by síce bolo zmeniť dotknuté rozhrania na abstraktné triedy, ale jedna trieda má výhodu najmä v tom, že všetky konštanty sú na jednom mieste a sú jednoducho k dispozícii pre modifikáciu.

4.5 Testovanie a refaktoring

Pre účely testovania bola vytvorená sada unitových testov v prostredí .NET. Testy sa snažia zachytiť celú funkcionalitu rozhrania XAPI a pomohli odhaliť mnoho chýb na klientskej, ale aj serverovej strane komunikácie. Bez použitia automatizovaných testov by bolo testovanie funkcionality veľmi obtiažne, naviac ich automatizácia umožnila zjednodušenie refaktoringu (revízia kódu, jeho zjednodušenie a vylepšenie bez zmeny vonkajšieho správania). Pomocou refaktoringu sa podarilo aplikáciu zprehľadniť a niektorými zmenami aj zrýchliť. Ide najmä o zavedenie triedy pre konfiguráciu, či zavedenie poolu pre spojenia. V následujúcich častich budú popísané niektoré najdôležitejšie problémy a nedostatky, ktoré boli pomocou refaktoru odstránené.

4.5.1 Samooprava strateného spojenia

Jedným z prvých odhalených problémov pri testovaní bol problém s naväzovaním spojenia. Išlo o pomerne nečakaný problém, ktorý sa však vyskytoval pomerne často, a najmä pri prenášaní veľkých súborov a dlhých operáciách s množstvom SOAP správ bolo rozhranie značne nestabilné. Presná príčina problému nebola úplne odhalená, ale zrejme súvisí s problémami pri prenose cez http protokol. Riešením, ktoré pomohlo tento problém takmer úplne neutralizovať, bolo zaviesť automatické opakovanie pokusu o spojenie po jeho zlyhaní s určitým čakacím časovým intervalom. V konfigurácii je možné nastaviť počet pokusov pre znovupripojenie a ich timeout. Toto opakovanie je implementované do najnižsej úrovne komunikácie, teda až na úroveň zasielania správ, to znamená, že aj pri prenose väčších súborov sa spojenie nepreruší.

4.5.2 Memory leak - Únik pamäti

Vývoj v jazyku C++ nesie vždy so sebou riziko nesprávneho odstraňovania objektov a tým nesprávneho uvoľňovania pamäti. Tento problém sa bohužiaľ prejavil naplno, keďže sa v serverovej časti využíva viacero knižníc a framework. Vo viacerých prípadoch nie je úplne jasné ako boli dané objekty vytvorené (halda alebo zásobník). Vzniklo teda viacero *memory-leakov*, z ktorých sa tie najväčšie podarilo odstrániť (niektoré sú zrejme bugy priamo frameworku Axis2/C). Odhalenie takýchto chýb je veľmi náročné, takže nie je možné povedať, že sú odstránené všetky. Táto chyba často spôsobovala neočakávaný pád aplikácie.

4.5.3 Pool spojení a cache pre URI

Medzi vylepšenia, ktoré umožnili zvýšenie výkonu aplikácia, patria hlavne pool spojení - *connection pool* a cache pre URI adresy. *Connection pool* vznikol ako snaha o ušetrenie prostriedkov pri pripájaní k serveru, ktoré je často časovo náročné. Základnou myšlienkou bolo vytvorenie trvalého spojenia na databázu, cez ktoré by sa zasielali požiadavky bez nutnosti po spustení opakovať vytváranie spojenia. Keďže XAPI umožňuje prácu s viacerými databázami naraz, nemohlo sa vytvoriť spojení neobmedzené množstvo. Vznikol teda takzvaný *pool*, ktorý má obmedzenú kapacitu (je možné ju nastaviť v konfigurácii) a pri prekročení limitu sa odstráni najdlhšie nepoužívane spojenie. Bohužiaľ sa naviazanie perzistentného spojenia ukázalo ako problematické - zrejme použitý SOAP framework má s týmto typom spojenia problémy, takže sa spojenia beztak vytvárajú jednorázovo. Avšak istý pozitívny efekt predsalen vznikol. Keďže pri parsovaní URI adresovanej kolekcie sa vždy overovala adresa servera, pool umožňuje potvrdiť jeho existenciu bez odoslania testovacej správy. To síce nie je tak efektné ako pôvodný zámer, ale po danej úprave bolo badateľné isté zvýšenie výkonu.

Na podobnom princípe funguje aj cache pre ukladanie URI kolekcií. Táto trieda vznikla z toho dôvodu, že vždy pri vytváraní inštancie triedy kolekcie je nutné rozparsovať URI adresu na typ databáze, adresu servera, názov databáze a názov kolekcie vrátane jej pozíciu v hierarchii kolekcií. Pri vytváraní lokálnych inštancií kolekcií sa často stávalo, že sa aj niekoľkokrát parsovalo rovnaké URI (prípadne URI s podobnou štruktúrou). To sa podarilo obmedziť zavedením cache, v ktorej sa vždy najskôr vyhľadá, či dané URI nebolo parsované (prípadne podobné znaky s inými URI) a vráti sa jeho rozparsovaná inštancia. Cache má podobne ako *connection pool* obmedzenú kapacitu a vymazáva nepotrebné záznamy.

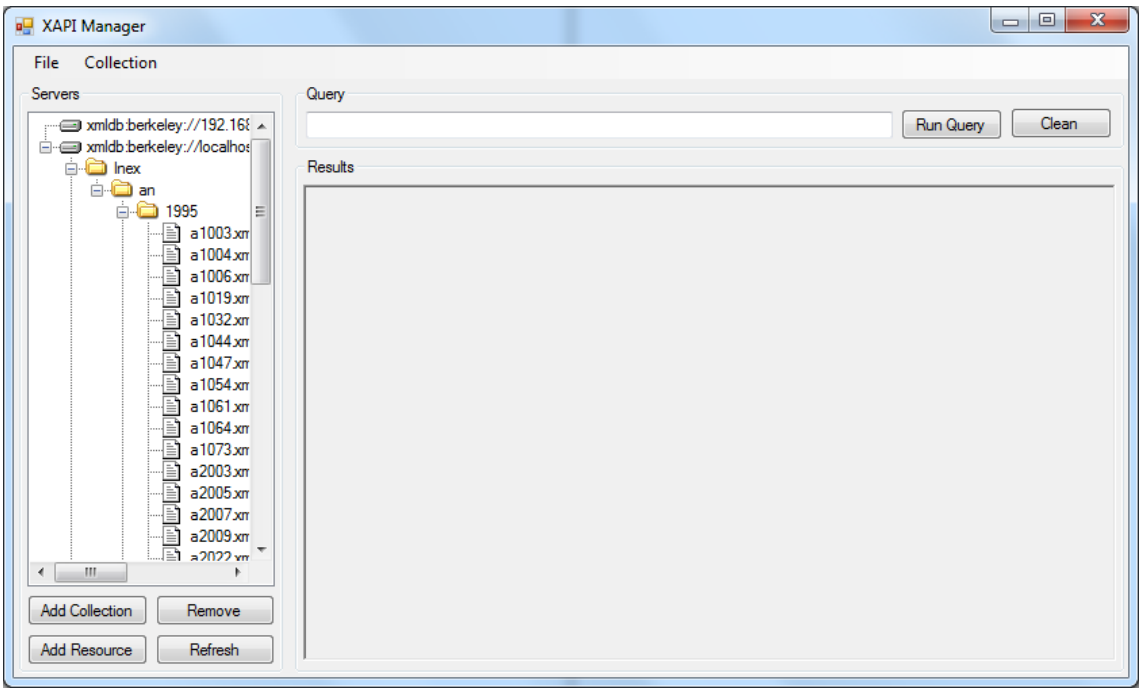
Obe triedy sú implementované pomocou návrhového vzoru singleton, ktorý vždy vytvára iba jednu inštanciu triedy, čo zaručuje konzistenciu údajov v rámci celého programu.

4.5.4 Konfigurácia

Počas vývoja sa v zdrojovom kóde nahromadilo množstvo rôznych hodnôt a limitov. Tieto hodnoty bolo nutné zozbierať a umiestniť na jedno miesto, kde by sa s nimi jednoducho manipulovalo a vyhľadávalo. Z tohto dôvodu vznikla trieda Config, ktorá je rovnako ako *connection pool* implementáciou návrhového vzoru singleton, keďže je vždy potrebná iba jedna inštancia. Ako bolo spomenuté, trieda obsahuje všetky nastavenia limitov (connection pool, limit URI cache) a ostatných hodnôt (počet a timeout opakovania pripojenia, maximálna veľkosť SOAP správy a podobne). V reálnej aplikácii sa konfigurácia nastaví po spustení programu z konfiguračného súbora, ako napríklad v ukážkovej aplikácii XAPI Manager sa načíta zo XML súbora config.xml.

4.6 Klientská aplikácia - XAPI Manager

Za účelom testovania a demonštrácie schopností implementácie rozhrania XAPI pre jazyk C# v prostredí .NET bola vytvorená sada ukážkových príkladov jeho použitia a jednoduchý manager databází XAPI Manager. Spomínaná sada ukážok je uložená na priloženom DVD v adresári *src/Ukazky pouzitia XAPI* a snaží sa pokryť všetky významné operácie, ktoré nové rozhranie poskytuje. Detailnejší popis rozhraní (podrobná JavaDoc dokumentácia) sa ďalšie ukážky je možné nájsť na stránkach iniciatívy XmlDb.org.



Obrázek 14: XAPI Manager

XAPI manager (obrázok 14) je, ako už bolo spomenuté jednoduchý manager, ktorý umožňuje správu databází, kolekcií, dokumentov a ich dotazovanie pomocou jazyka XPath. Aplikácia poskytuje prehľadné uživateľské rozhranie, kde je možné pripojiť sa k viacerým serverom naraz (môžu byť aj z rôznych typov databází). V ľavej časti okna je zobrazený strom kolekcií a dokumentov (zdrojov), ktorý sa rozbalí po pripojení k danej databáze. Pripojenie sa realizuje pomoucu dvojkliku na zvolený server alebo tlačidlom Connect.

Správa databází umožňuje vytvárať nové detské kolekcie (vyberie sa rodičovská kolekcia a zvolí sa meno) a ich vymazanie (koreňovú kolekciu nie je možné vymazať). Správa dokumentov funguje podobným spôsobom, pričom pre zobrazenie celého dokumentu je nutné ho vybrať dvojklikom a jeho obsah sa zobrazí v pravej časti okna. Rozhranie umožňuje vkladanie jednotlivých súborov, ale aj celých adresárov, kde je možné nastaviť, či sa budú vytvárať detské kolekcie podľa podadresárom. Je taktiež možné nastavit' generovanie unikátnych názvov dokumentov v prípade, že je riziko opakovania sa dokumentu s rovnakým názvom.

Spúšťať dotazy jazyka XPath je možné, buď na celé kolekcie alebo jednotlivé dokumenty, podľa toho čo je označené v strome kolekcií. Pri spúšťaní dotazov na kolekciu sa daný dotaz spustí aj na podkolekcie. Výsledky dotazov sú zobrazované v pravom okne a pridávajú sa vždy na koniec textu.Samotný textový blok sa vymazáva manuálne, kvôli možnosti porovnania výsledkov a pre zabránenie neúmyselnej straty informácií.

5 Porovnanie s ostatnými riešeniami

V záverečnej časti diplomovej práce sa pokúsime zmerať výkon rozhrania a porovnať ho s inými existujúcimi riešeniami. Zrejme najlepším spôsobom zistenia výkonu je jeho porovnanie s výkonom databáze priamo, t.j. bez vyvíjaného rozhrania, iba s použitím knižnice. Pre tento účel bola vytvorená séria testov, ktorá sa zameriava na rýchlosť vykonávania základných operácií databáze. Týmito operáciami sú :

1. Pripojenie k DB - rýchlosť schopnosti pripojiť sa k danej databáze.
2. Vytvorenie kolekcií - v prípade testu sa vytvorilo 5 podkolekcií.
3. Zoznam kolekcií - schopnosť získať zoznam detských kolekcií (5ks z predchodzej úlohy).
4. Vymazanie kolekcií - vymazanie detských kolekcií, 5 ks. z bodu 2.
5. Vloženie rozsiahleho súbora (100MB) - schopnosť vloženia veľkých súborov, testovací súbor bol vygenerovaný generátorom XMark [77] s faktorom 1.
6. Dotaz na súbor (XMark 100M) 1 - dotazovanie veľkých súborov s veľkým počtom výsledkov, konkrétne spustenie dotazu //people/person[profile[age and gender] and creditcard]/@id s očakávaným počtom výsledkov : 1645.
7. Dotaz na súbor (XMark 100M) 2 - spustenie dotazu //namerica/item[location = 'Canada']/name s očakávaným počtom výsledkov : 14.
8. Vloženie malých súborov (46 ks) - vloženie malých súborov z kolekcie INEX (INitiative for the Evaluation of XML Retrieval) [38].
9. Zoznam dokumentov v kolekcii
10. Získanie jedného dokumentu - získanie celého konkrétneho dokumentu z kolekcie.
11. Dotaz zameraný na malé dokumenty : INEX 1 - spustenie dotazu //fno s očakávaným počtom výsledkov : 46.
12. Dotaz priamo na určitý dokument : INEX 2 - spustenie dotazu //fno na konkrétny dokument s očakávaným počtom výsledkov : 1.
13. Vymazanie dokumentov (47ks) - vymazanie všetkých vložených dokumentov.
14. Rozvetvený dotaz na INEX (396 dokumentov) : INEX 3 - /article/bm/vt/p[b='Dan Geer' or @align='left'] s očakávaným počtom výsledkov : 38.
15. Rozvetvený dotaz na INEX (396 dokumentov) s relatívnou cestou : INEX 4 - //hdr[hdr2/pdt/yr='1998' and hdr1/ti='IEEE ANNALS OF THE HISTORY OF COMPUTING'] s očakávaným počtom výsledkov : 44.

V praxi nebolo možné porovnať priamo výkon s existujúcimi riešeniami rozhraní iných databází, keďže rozhrania sú naviazané na databázové operácie a ich výkon pri spúšťaní testov je priamo ovplyvnený efektivitou ukladania alebo dotazovanie danej databáze. Z tohto dôvodu sa toto porovnanie ani nesnaží porovnávať výkony čisto rozhraní, ale skôr sa zameriava na porovnanie výkonu rozhrania a databáze ako celku z pohľadu uživateľa. V praxi nás taktiež skôr ako výkon samotnej databáze alebo samotného rozhrania zaujíma skôr celkový výkon a miera využiteľnosti celku.

Pre porovnanie boli vybrané tieto natívne XML databáze :

- Sedna XML DB
- eXist
- Xindice

Všetky porovnávané riešenia majú implementované rozhranie XMLDB API (XAPI). Miernym problémom je, že ich XAPI je implementované v jazyku Java (vyvíjané XAPI je vôbec prvou implementáciou v prostredí .NET), čo môže mierne skresľovať výsledky, ale vo všeobecnosti sú prostredia Java a .NET na porovnateľnej výkonnostnej úrovni.

Dotaz	*Oracle C++*	*Oracle XAPI*	*Rozdiel*
Pripojenie k DB	—	192,6	—
Vytvorenie kolekcií	1271,3	1505,3	234,0
Zoznam kolekcií	—	83,3	—
Vymazanie kolekcií	93,7	442,7	349
Vloženie XMark	213802,4	1150955,1	937152,7
Dotaz XMark 1	1312,2	2069,1	756,9
Dotaz XMark 2	385,3	114,6	270,7
Vloženie INEX	88686,0	138418,8	49732,8
Zoznam dokumentov	78,1	161,6	83,5
Získanie dokumentu	88,6	359,6	271
Dotaz INEX 1	83,3	177,2	93,9
Dotaz INEX 2	92,5	192,8	100,3
Vymazanie dokumentov	321684,8	366699,8	45015,0
Dotaz INEX 3	442,6	250,0	192,6
Dotaz INEX 4	318,0	114,6	203,4

Tabulka 1: Výsledky porovnávacích testov[ms] mezi rozhraním a priamou manipuláciou

V tabuľke 1 sú uvedené výsledky porovnania v rámci zisťovania výkonu rozhrania[4]. Stĺpec *Oracle C++* obsahuje namerané hodnoty priameho prístupu a stĺpec *Oracle XAPI* časy prístupu cez XAPI. Z hľadiska zistenia výkonu rozhrania je najdôležitejší stĺpec *Rozdiel*, z ktorého je jasne vidieť oneskorenie, ktoré nastáva pri komunikácii cez vyvíjané rozhranie. Toto oneskorenie bolo očakávané a je spôsobené komunikáciou cez http protokol a spracovaním operácií na oboch stranách. Najväčšie spomalenie rozhrania oproti priamej manipulácii s knižnicou dochádza pri vkladaní veľmi veľkých dokumentov. Tento jav je spôsobený prenosom dokumentu *po častiach*, t.j. dokument sa prenáša po menších blokoch a vzniká teda veľké množstvo komunikácie, ktorá spomaľuje vkladanie a ostatné operácie, kde dochádza k hustejšej komunikácii. Hodnoty pre pripojenie k databáze a získanie zoznamu kolekcií neboli pre knižnicu vykonávané, pretože pripojením je myslené sieťové spojenie a to nemá pre priamu komunikáciu zmysel. V prípade zoznamu kolekcií zasa priama manipulácia nepodporuje ich stromovú hierarchiu (ako už bolo spomenuté, táto vlastnosť je v XAPI emulovaná), takže taktiež toto meranie nemá opodstatnenie. V prípade operácií, ktoré si vyžadujú menší počet odoslaných SOAP správ, ako napríklad vytvorenie kolekcií, či spúšťanie dotazov, je oneskorenie spôsobené rozhraním takmer zanedbateľné.

Dotaz	*Sedna*	*Exist*	*Xindice*	*Oracle XAPI*
Pripojenie k DB	73,3	5,1	21,8	192,7
Vytvorenie kolekcií	167	57,7	1323,3	1505,2
Zoznam kolekcií	15,6	10,0	15,7	83,3
Vymazanie kolekcií	166,3	177,2	541,7	442,7
Vloženie XMark	75453,6	414932,7	—	1150955,1
Dotaz XMark 1	3354,3	35199,2	—	2069,1
Dotaz XMark 2	828,0	1548,0	—	385,3
Vloženie INEX	9208,2	6433,3	55521,0	138418,8
Zoznam dokumentov	52,0	10,6	187,6	161,6
Získanie dokumentu	130,7	125,5	1463,2	359,6
Dotaz INEX 1	47,0	755,7	62,8	177,2
Dotaz INEX 2	46,3	52,3	62,8	192,8
Vymazanie dokumentov	253437,7	89917,9	2411,6	366699,8
Dotaz INEX 3	94,0	1219,0	3375,3	442,6
Dotaz INEX 4	295119,0	1063,6	3782,6	318,0

Tabulka 2: Výsledky porovnávacích testov[ms] s odstatnými riešeniami

Z hľadiska porovnania rozhrania databáze ako celku (tabuľka 2), je vyvíjané rozhranie relatívne konkurencie schopné a pokiaľ chceme databázu používať najmä na dotazovanie, tak dokonca viacerých prípadoch poráža ostatné riešenia. Okrem toho pokúka v prípade

[4]Testy boli vykonávané na počítači s konfiguráciou : Intel Core 2 Duo 1.4 GHz, 1,5 GB DDR2 RAM, Windows XP SP 2.

dotazovania najmä stabilný výkon, narozdiel od databáze Sedna XML DB, ktorá síce v absolútnych dotazoch exceluje, ale v prípade dotazov s relatívnou cestou (viď INEX 4) má ohromné problémy. Avšak pri operáciách ako vloženie dokumentov, či ich vymazanie Oracle XAPI, na ostatné riešenia dosť stráca. Za tento fakt je, ale do značnej miery zodpovedná aj implemenetácia databáze Oracle Berkeley XML DB, ako je vidieť v jej samostatných výsledkoch (najmä pri vkladaní malých dokumentov kolekcie INEX alebo vymazávaní vložených dokumentov).

Celkovo sú výsledky databází dosť rôznorodé a je z nich jasne vidieť, že používajú rôzne indexovacie techniky na rôznej úrovni detailnosti. Chýbajúce hodnoty databáze Xindice pre kolekciu XMark spôsobilo, že databáza príliš nepodporuje príliš veľké súbory a pri vkladaní 100 MB súbora nemala databáza dostatok pamäti (nestačili jej ani 2GB).

6 Záver

Účelom diplomovej práce bolo zozbierať informácie o existujúcich riešeniach softwarových komunikačných rozhraní pre natívne XML databáze, z nich jedno riešenie implementovať a pokusne napojiť na databázu. Počas zbierania informácií o rozhraniach a ich štandardoch sa ukázalo, že síce mnoho databází implementuje svoje vlastné riešenia, ale taktiež existuje aj niekoľko pokusov o zavedenie štandardu v tejto oblasti.

Pravdepodobne najpoužívanejším riešením je projekt iniciatívy XMLDB.org XAPI a z tohto dôvodu bolo toto rozhranie vybrané pre implementáciu v druhej časti. Keďže je rozhranie pôvodne navrhnuté pre jazyk Java, ide pravdepodobne o jednu z prvých implementácií v .NET. Pre ukážkové napojenie rozhrania na databázu bol vybraný systém Oracle Berkeley XML DB. Dôvodom výberu tohto zástupcu je, že sa jedná o embedded databázu, ide o knižnicu, ktorá manipuluje s dátami na lokálnom počítači. Jedná sa teda o ideálneho kandidáta pre implementáciu XAPI, ktoré je práve založené na sieťovej komunikácii a umožní Oracle databázu využívať aj prostredníctvom siete.

Rozhranie je rozdelené na 2 časti : serverovú a klientskú. Serverová časť je písaná v jazyku C++ a pre svoj beh využíva framework Axis2, ktorý umožňuje spustiť server samostatne alebo ho integrovať do servera Apache. Klientská časť implementuje špecifikáciu XMLDB.org a je napísaná v jazyku C#. Komunikácia prebieha pomocou zasielania SOAP správ, ktorých formát je definovaný v priloženom WSDL dokumente. Je teda možné využívať databázu aj bez klientskej časti, z ktoréhokoľvek programovacieho jazyka alebo prostredia umožňujúceho zasielanie SOAP správ, resp. http požiadavkov. To robí z vyvíjaného rozhrania univerzálny nástroj a umožňuje bez problémov vymeniť klientskú alebo serverovú časť bez väčších zmien. Umožňuje taktiež implementáciu ďalších klientských rozhraní bez obmedzenia.

Rozhranie bolo implementované v rozsahu stanovenom v špecifikácii. Implementácia teda spĺňa úroveň 0 z XMLDB.org a sú pridané podpory pre správu kolekcií, jazyka XPath a XUpdate. Jazyk XUpdate je podporovaný rozhraním, ale databáza Oracle Berkeley XML DB ho neobsahuje, a tak nie je táto funkcionalita v praxi využitá.

Počas záverečného testovania a porovnania s inými riešeniami sa ukázalo, že rozhranie dokáže konkurovať iným riešeniam z oblasti rozhraní pre natívne XML databáze. Rozhranie XAPI bolo vyvíjané s dôrazom najmä na spoľahlivosť prenosu a implementáciu, čo najväčšej funkcionality ponúkanej špecifikáciou iniciatívy XMLDB.org. Táto priorita bola v konečnom dôsledku aj splnená, už menší dôraz bol kladený na vyladenie výkonu rozhrania (najmä z časovej náročnosti implementácie kompletnej funkcionality), z toho dôvodu by ako námet na nadväzujúce práce, mohol byť refaktoring a vylepšenie výkonu rozhrania. V rámci ktorého by bolo možným riešením použitie iného SOAP servera, ktorý by lepšie vyhovoval potrebám rozhrania. Prípadne vývinúť vlastný server pre protokol TCP, ktorý by bol schopný vytvoriť s klientom trvalé spojenie (v prípade

Axis2 sa toto bohužiaľ nepodarilo) a spracovávať XML požiadavky v podobe SOAP správ.

Každopádne cieľ vyvinúť konkurencie schopné rozhranie, ktoré implementuje, čo najväčšie množstvo funkcionality potrebnej pre prácu s natívnymi XML databázami, bol dosiahnutý. Taktiež sa podarilo vytvoriť veľmi flexibilný nástroj pre komunikáciu s databázami, ktorého architektúra podporuje prakticky všetky programovacie jazyky a umožňuje jednoduchú modularitu prakticky na všetkých svojích úrovniach.

7 Literatúra

[1] N. Bruno and D. Srivastava and N. Koudas: Holistic Twig Joins: Optimal XML Pattern Matching. In Proceedings of the ACM International Conference on Management of Data, SIGMOD 2002, pages 310-321, ACM Press, 2002.

[2] B. Catania, A. Maddalena and A. Vakali, *XML Document Indexes: A Classification*, In Proc. IEEE INTERNET COMPUTING, September 2005, pp. 64 - 71.

[3] B. Catania et al., *Lazy XML Updates: Laziness as a Virtue of Update and Structural Join Efficiency*, In Proc. Int'l Conf. Management of Data (ACM Sigmod), ACM Press, 2005, pp. 515–526.

[4] C.W. Chung et al., *APEX: An Adaptive Path Index for XML Data*, In Proc. Int'l Conf. Management of Data (ACM Sigmod), ACM Press, 2002, pp. 121–132.

[5] R. Goldman and J. Widom, *DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases*,In Proc. Int'l Conf. Very Large Databases (VLDB 01), Morgan Kaufmann, 1997, pp. 436–445.

[6] H. Jiang et al., *XR-Tree: Indexing XML Data for Efficient Structural Joins*, In Proc. Int'l Conf. Data Eng. (ICDE 02), IEEE CS Press, 2002, pp. 253–263.

[7] M. Krátký, R. Bača: A Cost-based Join Selection for XML Twig Content-based Queries. In Proceedings of the Third International Workshop on Database Technologies for Handling XML Information on the Web, DataX 2008. EDBT, Nantes, France. Accepted, to appear in ACM DL. 2008.

[8] Michal Krátký, Radim Bača, Václav Snášel: On the Efficient Processing Regular Path Expressions of an Enormous Volume of XML Data. In Proceedings of 18th International Conference on Database and Expert Systems Applications, DEXA 2007. LNCS 4653, Springer Verlag. Regensburg, Germany.

[9] M. Krátký, J. Pokorný, V. Snášel: Implementation of XPath Axes in the Multidimensional Approach to Indexing XML Data. In Proceedings of Current Trends in Database Technology - EDBT 2004 Workshops, Heraklion - Crete, Greece. Springer-Verlag, Lecture Notes in Computer Science, Vol. 2368/2004, pp. 219-229, 2004.

[10] LAŠ, Michal. Testování efektivity přístupů pro indexování a dotazování XML dat. Ostrava, 2008. 45 s. Bakalárska práca. VSB-TU Ostrava.

[11] T. Milo and D. Suciu, *Index Structures for Path Expressions*, In Proc. Int'l Conf. Database Theory (ICDT 99), LNCS 1540, Springer-Verlag, 1999, pp. 277–295.

[12] P.R. Raw and B. Moon, *PRIX: Indexing and Querying XML Using Prüfer Sequences*, In Proc. Int'l Conf. Data Eng. (ICDE), IEEE CS Press, 2004, pp. 288–300.

[13] Hanan Samet: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, ISBN 978-0123694461, 2006.

[14] A. Silberstein et al., *BOXes: Efficient Maintenance of Order-Based Labeling for Dynamic XML Data*, In Proc. Int'l Conf. Data Eng. (ICDE), IEEE CS Press, 2005, pp. 285–296.

[15] Su-Cheng H, Chien-Sing L. Node Labeling Schemes in XML Query Optimization: A Survey and Trends. IETE Tech Rev [serial online] 2009 [cit. 2010-05-04];26:88-100. Dostupný z WWW: <http://tr.ietejournals.org/text.asp?2009/26/2/88/49086>

[16] H. Wang et al., *ViST: A Dynamic Index Method for Quering XML Data by Tree Structures*, In Proc. Int'l Conf. Management of Data (ACM Sigmod), ACM Press, 2003, pp. 110–121.

[17] H. Wang and X. Meng, *On the Sequencing of Tree Structures for XML Indexing*, In Proc. Int'l Conf. Data Eng. (ICDE), IEEE CS Press, 2005, pp. 372–383.

[18] W. Wang et al., *Efficient Processing of XML Path Queries Using the Disk-Based F&B Index*, In Proc. Int'l Conf. Very Large Databases (VLDB), Morgan Kaufmann, 2005.

[19] M. Yoshikawa, T.Amagasa, T. Shimura and S. Uemura: XRel: a Path-based Approach to Storage and Retrieval of XML Documents Using Relational Databases. In ACM Trans. Inter. Tech., Vol 1, Number 1, pages 110-141, ACM Press, 2001.

[20] Zhiyuan Chen and G.J. Korn and F. Koudas and N. Shanmugasundaram and J.D. Srivastava: Index Structures for Matching XML Twigs Using Relational Query Processors. In Proceedings of 13th International Conference on Data Engineering, ICDE 2005, IEEE Computer Society, 2005.

[21] Anatomy of an XML Database: Oracle Berkeley DB XML. An Oracle Whitepaper [online]. 2006, 09, [cit. 2010-04-21].
Dostupný z WWW: <http://www.oracle.com/webapps/dialogue/dlgpage.jsp?p_dlg_id=5039737&src=4945225&Act=4>

[22] An Introduction to the XML:DB API [online]. c2010 [cit. 2010-02-24].
Dostupný z WWW: <http://www.xml.com/pub/a/2002/01/09/xmldb_api.html>

[23] Apache Axiom - The XML Object Model [online]. c2004-2009 [cit. 2010-04-20].
Dostupný z WWW: <http://ws.apache.org/commons/axiom/>

[24] Apache Axis2/C - Apache Axis2/C - The Web Services Engine [online]. c2009 [cit. 2010-04-20].
Dostupný z WWW: <http://ws.apache.org/axis2/c/>

[25] Apache Xindice [online]. c2001-2007 [cit. 2010-02-22].
Dostupný z WWW: <http://xml.apache.org/xindice/>

[26] Apache XML Project - Xerces Parser [online]. c1999-2010 [cit. 2010-02-18].
Dostupný z WWW: <http://xerces.apache.org/>

[27] basex.org/ [Online] c2010 [cit. 2010-02-24]
Dostupný z WWW: <http://basex.org/>

[28] B-Trees [Online] [Dátum: 3. Máj 2008.]
http://www.bluerwhite.org/btree/

[29] Cascading Style Sheets [online]. c2010 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/Style/CSS/>

[30] Document Structure Description [online]. c1999-2005 [cit. 2010-02-18].
Dostupný z WWW: <http://www.brics.dk/DSD/>

[31] Document Type Definition [online]. c1999 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/TR/REC-html40/sgml/dtd.html>

[32] eXist-db Open Source Native XML Database [online]. c2010 [cit. 2010-02-24].
Dostupný z WWW: <http://exist.sourceforge.net/>

[33] Extensible HyperText Markup Language [online]. c2000 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/TR/xhtml1/>

[34] Extensible Markup Language [online]. c1996-2003 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/XML/>

[35] Extensible Stylesheet Language (XSL) Version 1.1 [online]. c2006 [cit. 2010-02-21].
Dostupný z WWW: <http://www.w3.org/TR/xsl/>

[36] HyperText Markup Language [online]. c1995-2007 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/MarkUp/>

[37] IBM - DB2 - Data server [online]. c2010 [cit. 2010-02-22].
Dostupný z WWW: <http://www-306.ibm.com/software/data/db2/>

[38] INEX [online]. c2007 [cit. 2010-04-30].
Dostupný z WWW: <http://inex.is.informatik.uni-duisburg.de/>

[39] Infonyte DB [Online] c2008 [cit. 2010-02-24]
Dostupný z WWW: <http://www.infonyte.com/en/products.html>

[40] Java JDBC Tutorial [online]. c2007-2008 [cit. 2010-02-22].
Dostupný z WWW: <http://www.jdbc-tutorial.com/>

[41] Mark Logic: MarkLogic Server [online]. c2010 [cit. 2010-02-24].
Dostupný z WWW: <http://www.marklogic.com/product/marklogic-server.html>

[42] Mark Logic XML Content Connector For Java (XCC/J) v4.1-5 [Online] c2003-2010 [cit. 2010-02-24]
Dostupný z WWW: <http://developer.marklogic.com/pubs/4.1/javadoc/overview-summary.html>

[43] Meggison Technologies: Simple API for XML [online]. c1999-2010 [cit. 2010-02-22].
Dostupný z WWW: <http://www.megginson.com/downloads/SAX/>

[44] Microsoft .NET Framework [online]. c2009 [cit. 2010-02-18].
Dostupný z WWW: <http://www.microsoft.com/NET/>

[45] MonetDB database system with XQuery front-end [online]. c1994-2009 [cit. 2010-02-24].
Dostupný z WWW: <http://monetdb.cwi.nl/XQuery/>

[46] MSXML[online]. c2010 [cit. 2010-02-18].
Dostupný z WWW: <http://msdn.microsoft.com/cs-cz/data/bb291077(en-us).aspx>

[47] Namespaces in XML 1.0 [online]. c2009 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/TR/REC-xml-names/>

[48] ODBC–Open Database Connectivity [online]. c2007 [cit. 2010-02-22].
Dostupný z WWW: <http://support.microsoft.com/kb/110093>

[49] OpenSSL: The Open Source toolkit for SSL/TLS [Online] c1999-2009 [cit. 2010-03-11]
Dostupný z WWW: <http://www.openssl.org/>

[50] Oracle Berkeley XML DB [online]. c2010 [cit. 2010-02-22].
Dostupný z WWW: <http://www.oracle.com/database/berkeley-db/xml/index.html>

[51] Oracle Database 11g [online]. c2010 [cit. 2010-02-22].
Dostupný z WWW: <http://www.oracle.com/us/products/database/index.htm>

[52] ozone [online]. c2010 [cit. 2010-02-24].
Dostupný z WWW: <http://www.ozone-db.org/frames/home/what.html>

[53] Prufer sequence [Online] [Dátum: 3. Máj 2008.]
http://en.wikipedia.org/wiki/Pr%C3%BCfer_sequence

[54] RELAX NG home page [online]. c2009 [cit. 2010-02-18].
Dostupný z WWW: <http://relaxng.org/>

[55] SAXON The XSLT and XQuery Processor [online]. c2010 [cit. 2010-02-22].
Dostupný z WWW: <http://saxon.sourceforge.net/>

[56] Schema for Object-Oriented XML [online]. c1999 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/TR/NOTE-SOX/>

[57] Sedna XML Database [online]. c2003-2009 [cit. 2010-02-24].
Dostupný z WWW: <http://modis.ispras.ru/sedna/index.html>

[58] SOAP Message Transmission Optimization Mechanism [online]. c2005 [cit. 2010-04-20].
Dostupný z WWW: <http://www.w3.org/TR/soap12-mtom/>

[59] SOAP specification [Online] c2004 [cit. 2010-02-24]
Dostupný z WWW: <http://www.w3.org/TR/soap/>

[60] SQL Server 2008 Overview, data platform, store data — Microsoft [online]. c2009 [cit. 2010-02-22].
Dostupný z WWW: <http://www.microsoft.com/sqlserver/2008/en/us/>

[61] Standard Generalized Markup Language [online]. c1995 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/MarkUp/SGML/>

[62] Tamino · The XML database [online]. c2010 [cit. 2010-02-22].
Dostupný z WWW: <http://www.softwareag.com/Corporate/products/wm/tamino/default.asp>

[63] W3C Document Object Model [online]. c1997-2005 [cit. 2010-02-22].
Dostupný z WWW: <http://www.w3.org/DOM/>

[64] W3C XML Schema [online]. c2000 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/XML/Schema>

[65] Web Services Architecture [online]. c2004 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/TR/ws-arch/>

[66] Web Services Description Language (WSDL) 1.1 [Online] c2001 [cit. 2010-02-24]
Dostupný z WWW: <http://www.w3.org/TR/wsdl>

[67] World Wide Web Consortium [online]. c2010 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/>

[68] Xalan-Java Version 2.7.1 [online]. c2006 [cit. 2010-02-22].
Dostupný z WWW: <http://xml.apache.org/xalan-j/index.html>

[69] XML/DBC API Tutorial [online]. c2003-2005 [cit. 2010-02-18].
Dostupný z WWW: <http://xquare.ow2.org/fusion/API%20Tutorial.html>

[70] XML:DB Initiative: API for XML Databases [online]. c2000-2003 [cit. 2010-02-18].
Dostupný z WWW: <http://xmldb-org.sourceforge.net/xapi/>

[71] XML:DB Initiative: SiXDML - Simple XML Data Manipulation Language [online]. c2000-2003 [cit. 2010-02-22].
Dostupný z WWW: <http://xmldb-org.sourceforge.net/sixdml/index.html>

[72] XML Path Language (XPath) [online]. c1999 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/TR/xpath>

[73] xqDoc Project – Quick Start [MarkLogic] [online]. c2000 [cit. 2010-02-18].
Dostupný z WWW: <http://xqdoc.org/qs_marklogic.html>

[74] XQuery 1.0 - An XML Query Language [online]. c2007 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/TR/xquery/>

[75] XQuery API for JavaTM (XQJ) [Online] c2010 [cit. 2010-02-24]
Dostupný z WWW: <http://jcp.org/en/jsr/summary?id=225>

[76] XQuery Update Facility 1.0 [online]. c2009 [cit. 2010-02-18].
Dostupný z WWW: <http://www.w3.org/TR/xquery-update-10/>

[77] XMark – An XML Benchmark Project [online]. c2001-2009 [cit. 2010-04-30].
Dostupný z WWW: <http://www.xml-benchmark.org/>

[78] XML-RPC Home Page [Online] c2004-2010 [cit. 2010-02-24]
Dostupný z WWW: <http://www.xmlrpc.com/>

[79] XSL Transformations (XSLT) [online]. c1999 [cit. 2010-02-21].
Dostupný z WWW: <http://www.w3.org/TR/xslt>

[80] XUpdate - XML Update Language [online]. c2000-2003 [cit. 2010-02-18].
Dostupný z WWW: <http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html>

A Obsah priloženého DVD

Obsah jednotlivý priečinkov priloženého DVD:

- `bin` - obsahuje spustiteľné verzie vyvíjaného software
  - `Klient – XAPI kniznica` - knižnica vo formáte dll, ktorú je nutné pridať do závislostí pri použití v aplikácii
  - `Server` - serverová časť rozhrania integrovaná vo frameworku Axis2/C
  - `XAPI Manager` - ukážková aplikácia, jednoduchý manager pre správu databází
- `src` - obsahuje zdrojové súbory vyvíjaného software
  - `Test vykonu – Oracle C++` - zdrojový kód aplikácie pre testovanie výkonu samostatnej C++ knižnice Berkeley databáze. (C++)
  - `Test vykonu – Oracle XAPI` - zdrojový kód aplikácie pre testovanie výkonu vyvinutého rozhrania. (C#)
  - `Test vykonu – Ostatne riesenia` - zdrojový kód aplikácie pre testovanie výkonu ostatných databází. (Java)
  - `Ukazky pouzitia XAPI` - ukážky použitia rozhrania XAPI v zdrojovom kóde. (C#)
  - `XAPI Klient` - zdrojový kód klientskej strany rozhrania, implementácia špecifikácie XMLDB.org. (C#)
  - `XAPI Klient – Unit testy` - unitové testy vytvorené pre otestovanie správnej funkčnosti rozhrania. (C#)
  - `XAPI Manager – ukazkova aplikacia` - zdrojový kód ukážkovej aplikácie. (C#)
  - `XAPI Server` - zdrojový kód serverovej časti rozhrania. (C++)
- `support` - obsahuje podporný software, konkrétne inštalačný súbor Oracle Berkeley XML DB databáze
- `text` - obsahuje text diplomovej práce vo formáte pdf

B Inštalácia serverovej časti

Vývoj serverovej časti rozhrania bol zameraný na multiplatformovosť (použitie v operačných systémoch rodiny Windows aj Linux). Aktuálne, pre potreby diplomovej práce bol, ale zatiaľ server nakompilovaný iba pre Windows. Inštalácia servera bola otestovaná na systémoch: Windows XP SP 2 a Windows 2003. Bohužiaľ na 64 bitový verziách bola pomerne nestabilná, takže sa zatiaľ odporúča týmto systémom vyhnúť.

Všetky potrebné súčasti inštalácie sú uvedené v prílohe na DVD. Testovacia prevádzka beží na serveri epe.vsb.cz, pričom pre testovaciu databázu je nutné zadať adresu epe.vsb.cz:9090/services/Test.

B.1 Postup inštalácie

1. Nainštalovať databázu Oracle Berkeley (binárny súbor bin\dbxml-2.5.13.msi), pričom cesta, kam bude systém nainštalovaný budeme označovať ako *OracleHome*.
2. Pridať cestu *OracleHome*\bin do systémovej premennej PATH.
3. Vytvoriť adresár *AxisHome*, kam sa má nainštalovať databázový server. Napríklad c:\database_server\.
4. Rozbaliť archív axis2.zip a obsah adresára axis prekopírovať do *AxisHome*.
5. Vytvoriť premennú prostredia AXIS2C_HOME a nastaviť jej hodnotu na cestu *AxisHome*. Napr. set AXIS2C_HOME=c:\database_server\.
6. Pridať cestu *AxisHome*\lib do systémovej premennej PATH.
7. Vytvoriťadresár *DbStorage*, kam sa budú fyzicky ukladať dáta z databáze. Napríklad c:\database_storage_directory\.
8. Upraviť súbor services\config.xml, kde je nutné prepísať hodnotu elementu *storage_folder* na aktuálnu hodnotu *DbStorage*.
9. Spustiť server pomocou *AxisHome*\bin\axis2_http_server.exe.

B.2 Vytvorenie novej databáze

1. Presunúť *AxisHome*\oracle_DB \db_service \#DB_NAME do *AxisHome*\services a premenovať ho na požadovaný názov novej databáze, ktorý bude odkazovaný ako *DbNazov*.
2. V adresári služby (novej databáze) premenovať súbor #DB_NAME.wsdl na *DbNazov*.wsdl.
3. Prekopírovať obsah adresára *AxisHome*\oracle_DB \root_collection_template do *DbStorage* a premenovať kopírovaný adresár z #DB_NAME na *DbNazov*.

4. Reštartovať server.

Pokiaľ prebehne všetko správne vytvorí sa databáza s prázdnou koreňovou a testovacou kolekciou (pomenovaná ako test, je ju možné ihneď vymazať, jej účel je čisto demonštratívny.)